

CUSTOMER

2017-11

# cXML solutions guide

Ariba Network



---

# Content

<b>Introduction to cXML solutions.</b>	<b>14</b>
cXML recommended usage.	14
About data mapping.	14
Maximum document size.	15
About cXML versions.	15
<b>cXML data conventions.</b>	<b>16</b>
Number format.	16
Money element.	17
Money format.	17
alternateCurrency and alternateAmount attributes.	18
UnitOfMeasurement element.	18
Name formats.	20
PostalAddress element.	20
Multiple lines in street element.	21
Country, State, and PostalCode elements.	21
TelephoneNumber element.	22
CountryCode element.	22
AreaOrCityCode and Number elements.	23
Date and Time format.	24
timestamp attribute.	24
Extrinsic elements.	24
<b>Document addressing and security.</b>	<b>26</b>
Overview of document security.	26
Document encryption.	26
cXML document authentication.	27
HTTPS connections.	27
HTTPS for buying organizations.	28
How to check URLs in Ariba Network accounts (Collaborative Invoicing PunchIn sites).	28
HTTPS for suppliers and service providers.	29
How to enable HTTPS on Web servers.	29
How to check URLs in your Ariba Network account.	29
cXML document authentication.	30
Available authentication methods.	30
Certificate authentication set up.	31
How to install your certificate on Ariba Network.	31

Changing your Ariba Network URLs. . . . .	32
Authenticate Ariba Network's Certificate. . . . .	32
Digital certificates. . . . .	32
Overview of digital certificates. . . . .	33
Client and server certificates. . . . .	33
Trusted certificate authorities. . . . .	34
Encryption strength. . . . .	34
Domain name in server certificates. . . . .	34
How to obtain a digital certificate. . . . .	34
About cXML credentials. . . . .	35
Supported ID domains. . . . .	36
Multiple credentials. . . . .	37
Case sensitivity. . . . .	38
Domain changes. . . . .	38
NetworkID and older versions of Ariba Buyer. . . . .	39
DigitalSignature element. . . . .	40
Test accounts. . . . .	40
Required credentials. . . . .	40
Ariba Buyer credentials. . . . .	41
Ariba Marketplace, Standard Edition. . . . .	43
Service providers. . . . .	44
Supplier credentials. . . . .	45
End points integration with Ariba Network adapters. . . . .	48
End point integration with Ariba Network adapters. . . . .	49
How to enable end point integration for outbound documents. . . . .	49
Enabling end point integration for Ariba Network Adapter for SAP NetWeaver. . . . .	50
End points integration in a multiple ERP setup. . . . .	50
Quick enablement. . . . .	51
Using purchase orders. . . . .	51
Using ICS invoices. . . . .	53
Using payment proposals. . . . .	55
Using CC invoices. . . . .	57
Using request for quotations. . . . .	59
Using tax IDs. . . . .	61
<b>Profile transaction. . . . .</b>	<b>63</b>
Using the Profile transaction. . . . .	63
Sending ProfileRequest documents. . . . .	63
Receiving ProfileRequest documents. . . . .	64
From Credential element. . . . .	64
Query frequency. . . . .	64
ProfileRequest document. . . . .	65

ProfileResponse document. . . . .	66
ProfileResponse implementation hints and limitations. . . . .	67
HTTPS URLs. . . . .	67
Options for OrderRequest. . . . .	67
Ariba Network test URLs. . . . .	68
<b>PunchOut site planning. . . . .</b>	<b>69</b>
PunchOut versus static catalogs. . . . .	69
Catalog hierarchy and requisition line items. . . . .	69
Deciding whether to use PunchOut. . . . .	70
PunchOut site implementation methodology. . . . .	70
Planning a PunchOut site. . . . .	71
Analysis of current and future states. . . . .	71
Outsourcing versus internal PunchOut site development. . . . .	72
Key participants for PunchOut integration. . . . .	72
Determining the level of PunchOut site support needed. . . . .	73
Designing the PunchOut site. . . . .	73
Supplier PunchOut site requirements. . . . .	74
Customer PunchOut site requirements. . . . .	74
Learning about cXML and PunchOut. . . . .	75
Developing the PunchOut site. . . . .	76
PunchOut message flow. . . . .	76
Extrinsics and supplier cookies. . . . .	78
Testing the PunchOut site. . . . .	78
Self-testing the PunchOut site on Ariba Network. . . . .	78
Testing the PunchOut site with customers. . . . .	78
Deploying the PunchOut site. . . . .	80
Configuration management. . . . .	80
Customer service. . . . .	80
Sanity check. . . . .	80
Becoming Ariba Ready certified. . . . .	80
Retrofitting an existing website. . . . .	81
Leveraging an existing site. . . . .	81
Quick retrofitting. . . . .	81
PunchOut for services. . . . .	82
Services exchange. . . . .	82
Create session. . . . .	82
Edit session. . . . .	83
Writing a PunchOut deployment guide. . . . .	83
PunchOut site connectivity overview. . . . .	83
PunchOut site authentication and identification. . . . .	83
PunchOut site required extrinsics. . . . .	84

PunchOut site content requirements/specification. . . . .	84
PunchOut site address information. . . . .	84
PunchOut site accounting structure. . . . .	85
PunchOut site commodity code description. . . . .	85
PunchOut site transactions supported. . . . .	86
<b>PunchOut transactions. . . . .</b>	<b>88</b>
PunchOut index catalog. . . . .	88
SupplierID element. . . . .	89
URL element. . . . .	89
Classification element. . . . .	89
punchoutLevel attribute. . . . .	89
Example PunchOut index file. . . . .	90
PunchOutSetupRequest document. . . . .	90
Credential elements. . . . .	91
UserAgent element. . . . .	92
PunchOutSetupRequest operation attribute. . . . .	92
BuyerCookie element. . . . .	93
Extrinsic element. . . . .	93
BrowserFormPost element. . . . .	94
SupplierSetup element. . . . .	94
SelectedItem element. . . . .	94
Example PunchOutSetupRequest document. . . . .	94
PunchOutSetupResponse document. . . . .	96
Overview of the PunchOutSetupResponse documents. . . . .	96
Status element. . . . .	96
StartPage URL element. . . . .	97
Example PunchOutSetupResponse document. . . . .	97
PunchOutOrderMessage document. . . . .	97
operationAllowed attribute. . . . .	98
cxml-base64 and cxml-urlencoded. . . . .	98
BuyerCookie element. . . . .	99
Total element. . . . .	99
ItemIn element. . . . .	100
ItemID element. . . . .	100
ItemDetail element. . . . .	101
Example PunchOutOrderMessage document. . . . .	103
PunchOut URL. . . . .	105
URL specified on the supplier's PunchOut site. . . . .	105
URL specified on Ariba Network. . . . .	106
URL specified in the supplier's index catalog (deprecated). . . . .	106
URL vs. SelectedItem elements. . . . .	107

Level 2 PunchOut. . . . .	107
User experience. . . . .	108
Item display and search. . . . .	108
Level 2 PunchOut mechanism. . . . .	108
Level 2 PunchOut format. . . . .	109
Level 2 PunchOut requirements. . . . .	110
CIF and cXML index content requirements. . . . .	110
Level 2 PunchOut site requirements. . . . .	111
Example index catalogs. . . . .	111
Store-level PunchOut index catalog. . . . .	111
Level 2 PunchOut index catalog. . . . .	112
Resulting PunchOutSetupRequest. . . . .	113
Level 2 PunchOut index catalog. . . . .	114
Shelf-level PunchOut. . . . .	115
Uploading index catalogs. . . . .	115
PunchOut session timeout. . . . .	116
ASP examples. . . . .	116
receivePunchoutSetupRequest.asp. . . . .	117
resolveXML.asp. . . . .	118
TCcXMLFormatDTime.asp. . . . .	119
<b>Purchase requisitions. . . . .</b>	<b>121</b>
Importing requisitions. . . . .	121
Importing a new requisition. . . . .	122
Updating a requisition. . . . .	124
Canceling a requisition. . . . .	125
<b>Purchase orders. . . . .</b>	<b>127</b>
OrderRequestHeader element. . . . .	127
orderID attribute. . . . .	127
orderDate attribute. . . . .	127
orderType attribute. . . . .	128
releaseRequired attribute. . . . .	128
effectiveDate attribute. . . . .	128
expirationDate attribute. . . . .	128
addressID attribute. . . . .	128
Name element. . . . .	129
DeliverTo element, line one. . . . .	129
DeliverTo element, line two. . . . .	129
Street element. . . . .	129
City element. . . . .	130
State element. . . . .	130

PostalCode element. . . . .	130
CarrierIdentifier element. . . . .	130
TransportInformation element. . . . .	131
Country isoCountryCode. . . . .	132
TelephoneNumber element. . . . .	132
Fax element. . . . .	132
Email element. . . . .	133
TermsofDelivery element. . . . .	133
URL element. . . . .	135
Tax element. . . . .	135
Modifications element. . . . .	136
ItemOut element. . . . .	139
Distribution element. . . . .	140
Accounting element. . . . .	140
ControlKeys element. . . . .	141
AccountingSegment element. . . . .	142
Segment element (deprecated). . . . .	143
Accounting fields in SAP Ariba Invoice Management. . . . .	144
Receiving types in SAP Ariba Invoice Management. . . . .	145
ReceivingType integration with external systems. . . . .	145
BlanketItemDetail element. . . . .	146
BlanketItemDetail in blanket purchase orders. . . . .	146
BlanketItemDetail in service purchase orders. . . . .	147
Masking values in blanket purchase orders and service purchase orders. . . . .	147
Header-level and line-level masking. . . . .	148
AmountBasedReceiving extrinsic. . . . .	148
ContingencyReceiving extrinsic. . . . .	149
InformationReceiving extrinsic. . . . .	149
Money currency attribute. . . . .	150
Modifications Element. . . . .	150
TermsofDelivery element. . . . .	150
PriceBasisQuantity element. . . . .	150
Tolerances element. . . . .	151
ScheduleLine element. . . . .	153
Ariba Network currency code mapping. . . . .	153
Cancel orders and change orders. . . . .	154
Cancel and change order requirements. . . . .	154
Cancel and change order requirements for buying organizations. . . . .	154
Cancel and change order requirements for suppliers. . . . .	155
Cancel and change orders and order status. . . . .	155
Purchase order matching. . . . .	155



Cancel orders. . . . .	155
Change orders. . . . .	156
Change orders in Ariba Buyer 8.0 and earlier. . . . .	157
Change orders in Ariba Buyer 8.1 and later. . . . .	157
Purchase order implementation hints and limitations. . . . .	158
Credential elements. . . . .	159
Contact roles. . . . .	159
Sold To Address and VAT ID. . . . .	159
Path routing. . . . .	160
Non-catalog items. . . . .	160
User-entered information for non-catalog items. . . . .	160
Enabling or disabling non-catalog orders. . . . .	161
Detecting non-catalog orders. . . . .	161
Breaking requisitions into multiple OrderRequests. . . . .	161
Item groups and grouped items in service orders. . . . .	161
Attachments. . . . .	162
Attachments on Ariba Network. . . . .	162
Attachment file names. . . . .	162
AttachmentOnline extrinsic. . . . .	163
Service sheets. . . . .	163
PCards. . . . .	164
Blanket purchase orders. . . . .	164
Subcontracting orders. . . . .	164
Incoterms on orders. . . . .	172
AribaNetwork.PaymentTermsExplanation extrinsic. . . . .	173
Schedule agreement releases. . . . .	173
ReleaseInfo. . . . .	174
ScheduleLineReleaseInfo. . . . .	175
agreementType. . . . .	175
ShipNoticePortion. . . . .	176
Planned and unplanned service line items. . . . .	176
Planned service line items. . . . .	176
Unplanned service line items. . . . .	179
Planned vs. unplanned service lines. . . . .	181
Extrinsic for clickable links. . . . .	181
Extrinsics for service periods. . . . .	182
Extrinsic for legacy purchase orders. . . . .	183
Changes introduced by Ariba Buyer 7.0.6. . . . .	183
External line numbers. . . . .	184
Service order with multi-level hierarchy. . . . .	184
Response documents. . . . .	190



Example response document. . . . .	190
Purchase order acknowledgments. . . . .	191
Other ways to acknowledge purchase orders. . . . .	192
Blanket purchase order acknowledgements. . . . .	192
Duplicate documents. . . . .	192
<b>Order confirmations and ship notices. . . . .</b>	<b>193</b>
Visibility of order status. . . . .	194
Visibility of order status for buying organizations. . . . .	194
Visibility of order status for requisitioners. . . . .	194
Visibility of order status for suppliers. . . . .	195
Ariba Network account configuration. . . . .	195
Buying organizations account configuration. . . . .	195
Suppliers account configuration. . . . .	195
Ariba Buyer 8.1 and later transaction flow. . . . .	196
Overview of ConfirmationRequest documents. . . . .	196
ConfirmationRequest element. . . . .	197
Implementation hints and limitations for ConfirmationRequest documents. . . . .	201
Example ConfirmationRequest. . . . .	204
OrderStatusRequest documents. . . . .	205
OrderStatusRequestHeader element. . . . .	205
OrderStatusRequestItem element. . . . .	206
Example OrderStatusRequest document. . . . .	206
ShipNoticeRequest documents. . . . .	207
ShipNoticeRequest implementation hints and limitations. . . . .	207
Example ShipNoticeRequest. . . . .	213
Canceling a ship notice using cXML. . . . .	214
Order confirmation and ship notice attachments. . . . .	215
Attachments on Ariba Network. . . . .	215
Attachment file names. . . . .	215
AttachmentOnline extrinsic. . . . .	215
<b>Service sheets. . . . .</b>	<b>217</b>
Visibility of service sheet order status. . . . .	218
Visibility of order status to buying organizations. . . . .	218
Visibility of order status to requisitioners. . . . .	218
Visibility of order status to suppliers. . . . .	218
Ariba Network account configuration. . . . .	218
ServiceEntryRequest configuration for buying organizations. . . . .	218
ServiceEntryRequest configuration for suppliers. . . . .	218
ServiceEntryRequest documents. . . . .	219
ServiceEntryRequest implementation hints and limitations. . . . .	219

Automatically-generated service sheets. . . . .	221
Service sheets and cancel and change orders. . . . .	223
Example ServiceEntryRequest. . . . .	223
Service sheet attachments. . . . .	226
ServiceEntryRequest status updates. . . . .	227
<b>Requests for quotation. . . . .</b>	<b>228</b>
Quote automation. . . . .	228
quoteRequest document. . . . .	229
quoteMessage document. . . . .	231
Example quoteRequest document. . . . .	232
Canceling requests for quotation. . . . .	235
SAP Ariba Sourcing integration with SAP ERP. . . . .	236
Sourcing integration with SAP Ariba Buying solutions. . . . .	240
QuoteRequest documents sent by SAP Ariba Buying solutions. . . . .	240
QuoteDataMessage documents sent from SAP Ariba Sourcing. . . . .	241
<b>Invoices and scheduled payments. . . . .</b>	<b>245</b>
Basic invoice functionality. . . . .	245
Purchase order matching. . . . .	246
Matching on Ariba Network. . . . .	246
Matching in Ariba Buyer. . . . .	247
Invoicing business rules. . . . .	247
Rules that affect PO-flip only. . . . .	248
Supplier visibility of business rules. . . . .	248
Numeric validation. . . . .	248
Invoice implementation hints. . . . .	249
eSignatures. . . . .	250
Supplier self-signed invoices. . . . .	250
IdReference element. . . . .	253
Contact roles. . . . .	254
Conversion of Segment to AccountingSegment. . . . .	255
Invoice payloadID attribute. . . . .	255
ServiceEntryItemReference element. . . . .	256
SerialNumber elements. . . . .	256
PaymentTerm element. . . . .	257
Remittance IDs. . . . .	257
Credit memos and line-item credit memos. . . . .	257
Inspection date attribute. . . . .	258
PriceBasisQuantity element. . . . .	258
itemType attribute. . . . .	259
compositeltemType attribute. . . . .	260

Tolerances element. . . . .	262
TaxDetail category. . . . .	263
Tax on shipping and special handling. . . . .	263
Allowances and charges. . . . .	265
French parafiscal taxes. . . . .	273
Withholding tax support. . . . .	278
Mandatory fields for countries with a VAT system. . . . .	279
Invoices for suppliers from Mexico. . . . .	285
Attachments with invoices. . . . .	289
Extrinsic elements. . . . .	290
Invoice element field lengths. . . . .	301
PDF invoice copy attachments. . . . .	302
Transmission of invoices to buying organizations. . . . .	303
Status and cancel invoices. . . . .	303
Invoice status. . . . .	303
Cancel invoices. . . . .	304
Example summary invoice. . . . .	304
Example header-level invoice. . . . .	308
Example blanket purchase order invoice. . . . .	311
Example of a complex buyer/supplier scenario. . . . .	313
Example purchase order: complex buyer/supplier scenario. . . . .	314
Example standard invoice: complex buyer/supplier scenario. . . . .	316
Example credit memo: complex buyer/supplier scenario. . . . .	321
Example debit memo: complex buyer/supplier scenario. . . . .	322
Example cancel invoice. . . . .	323
Scheduled payments. . . . .	325
For information only. . . . .	325
EFT payment. . . . .	325
Discount management. . . . .	325
Example scheduled payment. . . . .	327
<b>Receipts. . . . .</b>	<b>330</b>
ReceiptRequest routing. . . . .	330
cXML ReceiptRequest document. . . . .	330
ReceiptRequestHeader. . . . .	331
ReceiptOrder. . . . .	331
Example receipt document. . . . .	335
<b>Catalog upload transaction. . . . .</b>	<b>337</b>
Catalog upload transaction overview. . . . .	338
CatalogUploadRequest prerequisites. . . . .	338
CatalogUploadRequest document. . . . .	340

To element. . . . .	340
CatalogUploadRequest element. . . . .	340
Attaching your catalog. . . . .	343
Use a MIME envelope. . . . .	343
Compressing catalogs. . . . .	343
Example of sending a CatalogUploadRequest document. . . . .	343
Receiving the response. . . . .	345
Receiving later catalog status. . . . .	346
Email notification of catalog status. . . . .	346
URLPost notification of catalog status. . . . .	346
<b>Supply chain financing trades. . . . .</b>	<b>348</b>
<b>Get pending/data download transaction. . . . .</b>	<b>351</b>
Polling frequency. . . . .	352
Polling time. . . . .	352
Download all waiting documents. . . . .	352
Confirm the receipt of documents. . . . .	353
GetPendingRequest. . . . .	353
GetPendingResponse. . . . .	355
DataRequest. . . . .	356
DataResponse. . . . .	357
About invoice attachments. . . . .	358
lastReceivedTimeStamp and waiting documents. . . . .	358
Authenticating waiting documents. . . . .	359
<b>About SAP Ariba Supply Chain Collaboration. . . . .</b>	<b>360</b>
Consignment collaboration. . . . .	360
ConsignmentMovement example. . . . .	361
Business scenarios and assumptions for Forecast and Commit B2B. . . . .	363
<b>Contracts. . . . .</b>	<b>365</b>
ContractRequest. . . . .	365
ContractStatusUpdateRequest. . . . .	369
<b>cXML transformations on Ariba Network. . . . .</b>	<b>372</b>
Turning on transformations. . . . .	372
Moving extrinsics to the Contact element. . . . .	372
Transforming ItemOut extrinsics. . . . .	374
Consolidating ItemOut to OrderRequestHeader. . . . .	375
Mapping custom extrinsics to Ariba Network extrinsics. . . . .	375
Mapping extrinsics from cXML to EDI. . . . .	376
<b>Recommended coding systems. . . . .</b>	<b>377</b>

---

Commodity codes. . . . .	377
Currency codes. . . . .	378
Unit of measure codes. . . . .	378
Country codes. . . . .	379
Language codes. . . . .	379
Dialing codes. . . . .	380
<b>Revision history. . . . .</b>	<b>381</b>

---

# Introduction to cXML solutions

Ariba applications map data to and from documents using commerce eXtensible Markup Language (cXML). cXML is an open language defined by public Document Type Definitions (DTDs). These DTDs define cXML so that it is extremely flexible.

Ariba provides cXML solution guidelines and recommendation to supplement the general description of cXML provided by the *cXML User's Guide*.

## In this section:

[cXML recommended usage \[page 14\]](#)

[About data mapping \[page 14\]](#)

[Maximum document size \[page 15\]](#)

[About cXML versions \[page 15\]](#)

## cXML recommended usage

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

In particular you need to understand and work with the external cXML behavior of the following SAP Ariba applications:

- SAP Ariba Buying solutions
- Ariba Network

## About data mapping

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

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.

---

Use the Ariba guidelines and recommendations to implement consistent data between trading partners. They provide best practices for the conversion of back-end-system data to cXML to help prevent variation in the data transferred between trading partners.

## Maximum document size

The maximum size of cXML purchase orders, ship notices, order confirmations, and invoices is 40 MB or 10,000 lines. All other Ariba Network documents are limited to 4 MB or 1,000 lines.

The maximum size of attachments is configurable per customer. Customers can set the limit for attachments between 10 MB and 100 MB. The attachment limit applies to the total size of all attachments associated with the document. 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.

The Ariba Network limits the size of documents it can display to 1,000 line items. If a purchase order or invoice is larger than 1,000 lines, you can see only the header of the document when you open it. The line item details cannot be viewed. To download the complete document, you can open it and download the file in CSV format.

## About cXML versions

New versions of cXML are periodically released as the cXML standard is enhanced. cXML-enabled applications must be able to detect the cXML version of received documents and process those documents appropriately.

### Notification of cXML changes

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

### Additional references

- cXML Release Notes on <http://www.cxml.org>



---

# cXML data conventions

Use cXML data conventions to ensure data consistency among multiple trading partners. cXML provides data conventions for numbers, money elements, units of measurement, names, postal addresses, telephone numbers, data and time, and extrinsic elements.

## In this section:

- [Number format \[page 16\]](#)
- [Money element \[page 17\]](#)
- [UnitOfMeasurement element \[page 18\]](#)
- [Name formats \[page 20\]](#)
- [PostalAddress element \[page 20\]](#)
- [TelephoneNumber element \[page 22\]](#)
- [Date and Time format \[page 24\]](#)
- [Extrinsic elements \[page 24\]](#)

## Number format

The World Wide Web Consortium (W3C) XML Data Proposal defines data types used by cXML.

For more information about the W3C proposal, see (<http://www.w3c.org/TR/1998/NOTE-XML-data-0105>) .

W3C provides the following information about 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 restrictions and options 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 money element describes the data format to use for monetary values. It also provides attributes for specifying an alternate currency and amount.

### Example

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

Specify currency with three-letter ISO 4217 currency codes. For a list of these codes, see [Currency Codes \[page 378\]](#).

## Money format

cXML follows the W3C XML recommendation to use “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 different currencies, you might need to include alternate currency and amount information.

cXML provides the `alternateCurrency` and `alternateAmount` attributes for this describing alternate currency information.

### Example

```
<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 Buyer VAT ID and Supplier VAT ID \[page 282\]](#).

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

UNUOM Code	Meaning
RL	reel
PR	pair
PK	pack

For more information, see [Unit of Measure Codes \[page 378\]](#).

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’s 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.

```
<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. 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 be directed 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. It puts 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. When mail stop is included it should appear in the second `DeliverTo` element.

## Example

```
<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>
```

## Multiple lines in street element

Split `Street` elements where a comma might appear into additional `Street` elements. Ariba Network supports four `Street` elements per address.

For example, 1565 Charleston Rd., Suite 45 should appear as two `Street` elements:

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

For information about transformations performed on the `Street` element by Ariba Network see [cXML Transformations on Ariba Network \[page 372\]](#)

## Country, State, and PostalCode elements

A `PostalAddress` element requires that you provide a `Country` element with an `isoCountryCode` attribute. Without the `Country` element, a `PostalAddress` element is incomplete.

The `Country` value is a human-readable country name.

For the `isoCountryCode` attribute value, use ISO 3166-1 two letter codes. For more information, see [Country Codes \[page 379\]](#).

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 never contain a dash (-). 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.

### Note

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

## TelephoneNumber element

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

cXML `PhoneNumber` elements have the following format:

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

Example of a London phone number:

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

## 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 \[page 379\]](#). For ITU dialing codes, see [Dialing Codes \[page 380\]](#).



### Note

The ITU dialing code is not an ISO 3166-1 numeric country code (the United States has the value “840” in that system.)

**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.

### Note

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

## AreaOrCityCode and Number elements

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

Do not add “1” in the `AreaOrCityCode` element.

Do not use punctuation such as “(800) 555-5555” 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 not contain 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.

## timestamp attribute

The Ariba Network treats the `cXML@timestamp` attribute differently for different types of documents.

- Documents posted by Buyers (for example, Orders): The Ariba Network does not validate the `timestamp` value for documents posted by buyers. The `timestamp` can be any string, including a null string.
- Documents posted by Suppliers (for example, Invoices): The Ariba Network requires a valid `timestamp` value in ISO 8601 format for all documents posted by suppliers. By default, the Ariba Network uses the system date and time zone of the host (Pacific Standard Time) for the `timestamp` value. If the selected date is not the current date, the timestamp will be set to noon of that day.

## Extrinsic elements

Extrinsic elements define additional information that falls outside more standard commerce interactions, but which may be valuable to some organizations for their transactions. Ariba supports some extrinsics by default.

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 [cXML Transformations on Ariba Network \[page 372\]](#)

---

# Document addressing and 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.

## In this section:

- [Overview of document security \[page 26\]](#)
- [HTTPS connections \[page 27\]](#)
- [cXML document authentication \[page 30\]](#)
- [Digital certificates \[page 32\]](#)
- [About cXML credentials \[page 35\]](#)
- [Required credentials \[page 40\]](#)
- [End points integration with Ariba Network adapters \[page 48\]](#)
- [Quick enablement \[page 51\]](#)

## Overview of document security

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 \[page 27\]](#).

---

## 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 \[page 30\]](#).

## 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 cryptographic protocol called Transport Layer Security (TLS). TLS 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 TLS Web server certificate from a trusted certificate authority on your Web server. For more information about these certificates, see [Overview of Digital Certificates \[page 33\]](#).
- **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 communications.

### Related Information

[HTTPS For Buying Organizations \[page 28\]](#)

[HTTPS For Suppliers and Service Providers \[page 29\]](#)

---

## HTTPS for buying organizations

You must configure Ariba Buyer and SAP Ariba Sourcing to send documents to Ariba Network URLs prefaced with HTTPS.

Ariba Buyer and SAP 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 TLS certificate.

For complete information on setting up HTTPS communication between SAP 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 in web addresses.

## How to check URLs in Ariba Network accounts (Collaborative Invoicing PunchIn sites)

If you use a Collaborative Invoicing PunchIn site, check that your URLs use HTTPS.

### Procedure

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

### Results

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 \[page 63\]](#).

---

## HTTPS 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.

### How to enable HTTPS on Web servers

Web servers used by your cXML applications must support HTTPS.

#### Procedure

1. Ensure that your Web server is capable of supporting Transport Layer Security (TLSv1, TLSv1.1, or TLSv1.2).  
Support for TLSv1 will end on December 31.
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 \[page 34\]](#).

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.

### How to check URLs in your Ariba Network account

Set up your ProfileRequestURL, PunchOutSetupRequest, and OrderRequest URLs to support HTTPS.

#### Procedure

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 \[page 63\]](#).



- 
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, SAP 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, or digital certificate.

- **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).

#### Note

SAP Ariba Cloud Integration Gateway does not support shared secret in the cXML payload. If suppliers send shared secret, SAP Ariba Cloud Integration Gateway will remove this information before sending the document to Ariba Network.

- **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 TLS 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.

## How to install your certificate on Ariba Network

To set up certificate authentication on Ariba Network you must install a digitally signed certificate.

### Procedure

1. Obtain a signed digital certificate from a Certificate Authority trusted by SAP Ariba.
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.

#### Note

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**.

### Next Steps

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.

## Changing 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.

### Note

Suppliers cannot use self-signed certificates, but can use HTTPS to connect to SAP Ariba Cloud Integration Gateway for cXML.

## Authenticate 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.

### Related Information

[Overview of Digital Certificates \[page 33\]](#)

[About Client and Server Certificates \[page 33\]](#)

[Trusted Certificate Authorities \[page 34\]](#)

[Encryption Strength \[page 34\]](#)

[Domain Name in Server Certificates \[page 34\]](#)

[Obtaining a Digital Certificate \[page 34\]](#)

---

## 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.

### **i** 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

Server certificates are used for TLS, which is required for accepting HTTPS connection requests. These certificates are also called Web server certificates, secure server certificates, and secure server IDs. Client certificates are used for “certificate-based cXML document authentication.”

For more information about HTTP, see [HTTPS Connections \[page 27\]](#). For more information about cXML authentication, see [cXML Document Authentication \[page 30\]](#).

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 TLS and certificate based authentication, ask your certificate authority to issue one that is both a server and a client certificate.

---

## Trusted certificate authorities

You can use a signed digital certificate issued by any issuing organization. It must reference a root certificate from a Certificate Authority (CA) trusted by SAP Ariba. To get information about the CAs currently trusted by SAP Ariba or to register a CA to be trusted by SAP Ariba, contact SAP Ariba Customer Support.

## Encryption strength

SAP 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 TLS 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, a fully qualified DNS domain name takes the form “https://www.example.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.

## How to obtain a digital certificate

Obtain a signed digital certificate from a Certificate Authority trusted by SAP Ariba.

### Procedure

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**.

---

Request a server certificate for TLS 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 [About Client and Server Certificates \[page 33\]](#)

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 -----
```

## Next Steps

- To use the certificate to enable HTTPS, see [HTTPS connections \[page 27\]](#).
- To use the certificate for cXML document authentication, see [cXML document authentication \[page 30\]](#).

# About cXML credentials

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

## Related Information

[Supported ID domains \[page 36\]](#)

[Multiple credentials \[page 37\]](#)

[Case sensitivity \[page 38\]](#)

[Domain changes \[page 38\]](#)

[NetworkID and older versions of Ariba Buyer \[page 39\]](#)

[DigitalSignature element \[page 40\]](#)

[Test accounts \[page 40\]](#)

---

## Supported ID domains

SAP Ariba applications use a number of 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 <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 a buyer's 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 SAP Ariba Procurement solutions and buyers using Ariba Document 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 buyer's 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 SAP Ariba Procurement solutions and buyers using Ariba 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 SAP Ariba Procurement solutions only) An automatically generated supplier location private ID (called the Ariba Network Private ID in SAP Ariba Procurement solutions). This ID is generated in SAP Ariba Procurement 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 SAP 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 \[page 51\]](#).



## 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.

### 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>
```

### Example

Here is an example of the `To` credential from SAP Ariba Procurement 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">
          </IdReference>
        </IdReference>
      </Identity>
    </Credential>
  </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.

### Note

SAP Ariba Cloud Integration Gateway does not support multiple credentials in the header of the message, and the sender or receiver credentials must be ANIDs.

## 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 `SSPPPrivateID` 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 `SSPPPrivateID`) 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 using earlier versions of Ariba Buyer

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 with customers using earlier versions of Ariba Buyer

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 `ToCredential`.)

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. SAP Ariba applications do not use this element.

### **i** Note

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 \[page 30\]](#).

## 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.

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.

### 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.

## Related Information

[Ariba Buyer Credentials \[page 41\]](#)

[Ariba Marketplace, Standard Edition \[page 43\]](#)

[Service Providers \[page 44\]](#)

[Supplier Credentials \[page 45\]](#)

## Ariba Buyer credentials

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 \[page 39\]](#).

### 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

Element	Content
From	Ariba Buyer
To	Ariba Network
Sender	Ariba Buyer

### 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

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

Element	Content
Sender	Marketplace

## Example

```
<From>
  <Credential domain="NetworkID">
    <Identity>AN6565656565</Identity> <!-- identity of the marketplace -->
  </Credential>
</From>
<To>
  <Credential domain="AribaNetworkUserId">
    <Identity>admin@ariba.com</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="NetworkID">
    <Identity>AN6565656565</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

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>
```

## Supplier credentials

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>AN1000000123</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>AN0100000001</Identity>
  </Credential>
</From>
<To>
  <Credential domain="NetworkID">
    <Identity>AN1000000123</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="NetworkID">
    <Identity>AN0100000001</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>AN10000000123</Identity>
  </Credential>

```

```

</From>
<To>
  <Credential domain="NetworkID">
    <Identity>AN01000000456</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="NetworkID">
    <Identity>AN10000000123</Identity>
    <SharedSecret>abracadabra</SharedSecret>
  </Credential>
  <UserAgent>Our order-receiving system 3.0</UserAgent>
</Sender>

```

## End points integration with Ariba Network adapters

The cXML Adapter supports end points integration for the following solutions for Ariba Network Adapter for Oracle® Fusion Middleware, Ariba Network Adapter for SAP NetWeaver®, Ariba Network Transport Adapter, SAP Ariba Buying and Invoicing™, and SAP Ariba Buying™.

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. SAP 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 SAP Ariba procurement solution in Ariba Network, be sure to work with your SAP 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>sharedsecret</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.

### **i** Note

In a single ERP configuration, you do not need to include in the System ID in the <From> credential.

## End point integration with Ariba Network adapters

A new configuration parameter, `endpoint` added to the channel configuration file captures the end point information you configure in Ariba Network for your external ERP system. This parameter is optional. Set it only if you have configured an end point for your external ERP system in Ariba Network.

### **i** Note

This procedure is applicable only to the Ariba Network Adapter for Oracle Fusion Middleware and Ariba Network Transport Adapter.

## How to enable end point integration for outbound documents

You can enable end point integration for outbound documents.

### Procedure

1. Open the channel configuration file for the respective adapter.
2. Specify values for the relevant parameters.
3. Save the channel configuration file.

# Enabling 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**, `doubleRPEndPoint1` is the value that you have configured in the communication channel in SAP NetWeaver or channel configuration file.

## **i** 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.

## Example

```
<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>welcomela</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.

## 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 SAP 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 SAP 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 \[page 51\]](#))
- Invoices (see [Using ICS invoices \[page 53\]](#))
- Payment proposals (see [Using payment proposals \[page 55\]](#))
- CC Invoices (invoices sent from the Ariba procurement solution or the ERP system) (see [Using CC invoices \[page 57\]](#))
- Request for Quotations (see [Using request for quotations \[page 59\]](#))
- Tax IDs (see [Using tax IDs \[page 61\]](#))

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

```

## How buying organizations specify supplier IDs

Buying organizations use the `PrivateId` domain in the `To` `Credential` to identify the supplier. Buying organizations using SAP Ariba Procurement solutions also use the `VendorID`, `VendorSiteID`, and `SSPPriateID` domains.

Ariba Network assigns the supplier a `NetworkID` (AN-ID), but the buying organization can continue to use the [other domains \[page 36\]](#) in subsequent documents.



## How buying organizations specify 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 provide the `name="routing"` 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.

If the buying organization uses the Ariba Network light account capability to enable suppliers, the supplier registers a light account on Ariba Network. If the buying organization doesn't use the light account method, the supplier registers a full-use account.

### Note

Quick enablement purchase orders sent to light accounts aren't subject to supplier fees. Quick enablement purchase orders sent to full-use accounts are subject to supplier fees. For full-use accounts, if suppliers don't register and pay the necessary fees if they become chargeable, Ariba Network rejects purchase orders addressed to them. For more information about supplier fee thresholds, see the [subscriptions and pricing page](#).

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.

## 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 Network 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>Beamont</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>
```

## How service providers specify 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 SAP Ariba Customer Support to take ownership of quick enablement accounts created through invoicing.

If the buying organization uses the Ariba Network light account capability to enable suppliers, the supplier registers a light account on Ariba Network. If the buying organization doesn't use the light account method, the supplier registers a full-use account.

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.

### Example

```
<?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>

```

## How buying organizations specify supplier IDs

Buying organizations identify suppliers with the `PrivateID` domain in the `To Credential`. Buying organizations using SAP Ariba Procurement 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 \[page 36\]](#) in subsequent documents.

## How buying organizations specify 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 purchase orders that do not provide the `name="routing"` attribute.

### Note

- 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 can't send additional payment proposals to this supplier through Ariba Network.
- Ariba Network supports 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.

## How private suppliers take 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.

If the buying organization uses the Ariba Network light account capability to enable suppliers, the supplier registers a light account on Ariba Network. If the buying organization doesn't use the light account method, the supplier registers a full-use account.

Once the private supplier logs in and takes ownership of the account, the supplier can view the payment proposal. All subsequent notifications for 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 supplier fees. For more information about supplier fee thresholds, see the [subscriptions and pricing page](#).

## Using CC invoices

A CC invoice is a cXML copy request document that contains an attached invoice sent from SAP Ariba Buying solutions or the ERP system. The quick enablement through CC invoices must be enabled on the buyer's Ariba Network account.

Ariba Network stores a copy of the invoice originating from the ERP system. 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.

### Example

```
<?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>

```

## How buying organizations specify supplier IDs

Buying organizations identify suppliers with the `PrivateID` domain in the `To Credential`. Buying organizations using SAP Ariba Procurement 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 \[page 36\]](#) in subsequent documents.

## How buying organizations specify 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 purchase orders that do not provide the `name="routing"` attribute.

### **i** Note

- Ariba Network does not support the `fax` element for CC invoices sent from the SAP Ariba Buying solutions 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.

## How private suppliers take 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.

If the buying organization uses the Ariba Network light account capability to enable suppliers, the supplier registers a light account on Ariba Network. If the buying organization doesn't use the light account method, the supplier registers a full-use account.

Once the private supplier logs in and takes ownership of the account, the supplier can view the CC invoices sent from the SAP Ariba Buying solutions or the ERP system. 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 SAP Ariba Buying solutions or the ERP system are subject to supplier fees only when the private supplier has taken ownership of a full-use account on Ariba Network. Private suppliers who have not yet taken ownership of their account can continue to receive CC invoices without being charged for them. For more information about supplier fee thresholds, see the [subscriptions and pricing page](#).

## Using request for quotations

Procurement applications add the `Correspondent` element in the `quoteRequest` document to create Ariba Network accounts for suppliers.

### Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/schemas/cXML/1.2.025/Quote.dtd">
<cXML payloadID="123@jnk" timestamp="2013-11-19T11:50:11+00:00" version="1.2.025">
  <Header>
    <From>
      <Credential domain="NetworkId">
        <Identity>AN02000533043</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkId">
        <Identity>AN02000533043</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN02000533043</Identity>
        <SharedSecret>sharedsecret</SharedSecret>
      </Credential>
      <UserAgent>Procurement App 2.0</UserAgent>
    </Sender>
  </Header>
  <Request deploymentMode="production">
    <QuoteRequest>
      <QuoteRequestHeader>
        requestID="2021.2215" requestDate="2014-03-06T23:25:01.000-07:00"
```

```

type="new" openDate="2014-03-10T23:13:49.000-07:00" currency="USD"
xml:lang="en" loseDate="2014-03-17T00:00:00.000-07:00"
quoteReceivingPreference="winningOnly" >
<SupplierSelector matchingType="invitationOnly">
  <SupplierInvitation>
    <OrganizationID>
      <Credential domain="VendorID">
        <Identity>2210</Identity>
      </Credential>
    </OrganizationID>
    <Correspondent>
      <Contact>
        <Name xml:lang="en">New Inv Inc</Name>
        <PostalAddress>
          <DeliverTo>922 West</DeliverTo>
          <Street>15th Avenue</Street>
          <City>California</City>
          <State>CA</State>
          <PostalCode>93504</PostalCode>
          <Country isoCountryCode="US"></Country>
        </PostalAddress>
        <Email>vp@abc.com</Email>
        <Phone>
          <TelephoneNumber>
            <CountryCode isoCountryCode="US"></CountryCode>
            <AreaOrCityCode>ca</AreaOrCityCode>
            <Number>2342344</Number>
          </TelephoneNumber>
        </Phone>
        <URL></URL>
      </Contact>
    </Correspondent>
  </SupplierInvitation>
</SupplierSelector>
<Description xml:lang="" >Quote for AS1</Description>
<ShipTo>
  <Address isoCountryCode="US" addressID="409" >
    <Name xml:lang="en" >P2- Los Angeles</Name>
    <PostalAddress name="P2- Los Angeles" >
      <Street>1022 Sepulveda Blvd</Street>
      <Street>Suite 1000Address2</Street>
      <Street>Address3</Street>
      <City>El Segundo</City>
      <State>California</State>
      <PostalCode>90245</PostalCode>
      <Country isoCountryCode="US" >United States</Country>
    </PostalAddress>
    <Phone>
      <TelephoneNumber>
        <CountryCode isoCountryCode="US" />
        <AreaOrCityCode/>
        <Number/>
      </TelephoneNumber>
    </Phone>
    <Fax>
      <TelephoneNumber>
        <CountryCode isoCountryCode="US" />
        <AreaOrCityCode/>
        <Number/>
      </TelephoneNumber>
    </Fax>
  </Address>
</ShipTo>
</QuoteRequestHeader>
</QuoteRequest>
</Request>

```



---

## How buying organizations use supplier organization IDs

Buying organizations use the `Correspondent` element to store the contact information of the supplier. Ariba Network uses the `OrganizationID` in the `quoteRequest` document to identify if the supplier is a new or existing supplier.

For SAP, Oracle, and PeopleSoft domains, buying organizations identify suppliers using the `NetworkID` or `VendorID`, `VendorSiteID`, and `SiteAuxID` domains.

If the supplier does not exist on Ariba Network, the contact information is used to send the quote letter inviting the supplier to register and respond to the request for quote (RFQ) posting on SAP Ariba Discovery.

For more information about `Credential` domains, see [Supported ID domains \[page 36\]](#).

## How buying organizations specify email routing

Buying organizations can include the supplier's email address in the `Contact` element to instruct SAP Ariba Discovery where to route the quote letter invitation to the supplier.

### **i** Note

- Ariba Network supports cXML version 1.2.025 and later for quick enablement through requests for quotation.
- Ariba Network supports attachments and comments in the RFQ document.

## How SAP Ariba Discovery invites private suppliers to take ownership of accounts

If the supplier does not exist, SAP Ariba Discovery sends the quote automation invitation to the private supplier to complete the registration process.

Once the private supplier logs in and takes ownership of the account, the supplier can log in to view the request for quote (RFQ) posting from the buyer. All subsequent notifications for the RFQ document are sent to the supplier through email.

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

## Using tax IDs

Tax IDs are applicable only to buyers and suppliers that are set up for quick enablement through Tax IDs using XML invoices sent from Brazil. To enable this feature, a buyer must contact SAP Ariba Customer Support and configure their Tax IDs.

Once configured, buyers can view the TaxIDs for their organization from their Ariba Network account.

---

A supplier sends an NFe, CTe, and CCe invoice to a buyer through Ariba Network in an XML format. These invoices can be sent through an email or posted to Ariba Network where they are converted to the cXML file. For more information on sending these invoices, please contact SAP Ariba Customer Support.

**i Note**

- Ariba Network accepts and processes the NFe invoices only if the buyer has a configured TaxID on Ariba Network.
- Once the NFe invoice is successfully processed on Ariba Network, suppliers can send the TaxID in the domain for the cXML credential element.

## How Ariba Network uses TaxID to identify buyers

Ariba Network identifies buyers using the TaxID domain.

When a supplier sends an NFe invoice, Ariba Network verifies that the Tax ID for the buyer is configured on Ariba Network. If the Tax ID exists, Ariba Network also verifies that the buyer has an active trading relationship with the supplier using the information on the NFe invoice.

Ariba Network creates a new private supplier if the supplier does not exist on Ariba Network. Buyers can invite the supplier (vendor) by starting supplier enablement for the vendor on Ariba Network.

**i Note**

Buyers must verify the supplier email address before starting supplier enablement for them. For more information about supplier enablement, see *Enabling suppliers on Ariba Network*.

## How suppliers can take ownership of accounts

Suppliers can use the trading relationship invitation letter to log in to Ariba Network and register as a new supplier or use an existing account on Ariba Network.

Quick enablement through Tax ID is not subject to supplier fees. For more information about supplier fee thresholds, see the [subscriptions and pricing page](#).

For more information about invoices from Brazilian suppliers, see the *Ariba Network guide to invoicing*.

---

# 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.

## In this section:

[Using the Profile transaction \[page 63\]](#)

[ProfileRequest document \[page 65\]](#)

[ProfileResponse document \[page 66\]](#)

[ProfileResponse implementation hints and limitations \[page 67\]](#)

## Using the Profile transaction

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 SAP 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 <code>ProfileRequest</code> 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 <code>ProfileRequest</code> 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.
SAP Ariba Sourcing	Posting <code>ProfileRequest</code> documents to SAP 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.

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

A cXML application's transaction URLs can periodically change due to requirements of 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></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 payloadID="1189597956670-4363355987563302335@10.10.13.240"
timestamp="2007-09-12T04:52:36-07:00" >
  <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>
</cXML>

```

## ProfileResponse document

A ProfileResponse document is returned by Ariba Network in response to a ProfileRequest.

### Example

```

<?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://svccrms.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

## ProfileResponse implementation hints and limitations

For successful creation of `ProfileResponse` documents use HTTPS URLs, select the correct options for `OrderRequest`, and test your implementation before deployment.

### Related Information

[HTTPS URLs \[page 67\]](#)

[Options for OrderRequest \[page 67\]](#)

[Ariba Network Test URLs \[page 68\]](#)

## HTTPS URLs

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

For more information about HTTPS, see [HTTPS Connections \[page 27\]](#).

## Options for OrderRequest

The cXML specification provides `attachments` and `changes` 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%E2%80%A8>; 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%E2%80%A8>; 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%E2%80%A8>; 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.



---

# PunchOut site planning

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.

By following the methodology suggested here, you can plan and promote a solution for one customer than can scale easily to additional customers.

## In this section:

- [PunchOut versus static catalogs \[page 69\]](#)
- [PunchOut site implementation methodology \[page 70\]](#)
- [Planning a PunchOut site \[page 71\]](#)
- [Designing the PunchOut site \[page 73\]](#)
- [Developing the PunchOut site \[page 76\]](#)
- [Testing the PunchOut site \[page 78\]](#)
- [Deploying the PunchOut site \[page 80\]](#)
- [Becoming Ariba Ready certified \[page 80\]](#)
- [Retrofitting an existing website \[page 81\]](#)
- [PunchOut for services \[page 82\]](#)
- [Writing a PunchOut deployment guide \[page 83\]](#)

## PunchOut versus static catalogs

Ariba Buyer displays both static and PunchOut catalog items in the local catalog hierarchy, enabling users to choose items and insert them in requisitions. Static catalog content resides completely in Ariba Buyer, and 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

Development of a PunchOut site can be time-consuming and costly. Consider that 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 your customers to see whether they allow it.

## PunchOut site implementation methodology

Implementing a PunchOut site spans from initial evaluation of your existing e-commerce system to development, and then 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 for creating a PunchOut site are:

- [Planning a PunchOut Site \[page 71\]](#)
- [Designing the PunchOut Site \[page 73\]](#)
- [Developing the PunchOut Site \[page 76\]](#)
- [Testing the PunchOut Site \[page 78\]](#)
- [Deploying the PunchOut Site \[page 80\]](#)
- [Becoming Ariba Ready Certified \[page 80\]](#)

---

## Planning a PunchOut site

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 states

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.

### Evaluate your current site

Evaluate your current site for e-commerce, XML integration, electronic order processing, connection with your accounting system, integration with customers' control data, and ability to deliver custom pricing.

- 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?

### Plan for the PunchOut site

Evaluate your process flow from shopping to order placement and fulfillment, how much customization you want to provide per company, and how you will model a "shopping cart," among other considerations.

- 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 PunchOut site development

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 for PunchOut integration

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:  
<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 [http://www.ariba.com/suppliers/supplier\\_consulting.cfm](http://www.ariba.com/suppliers/supplier_consulting.cfm).

## Determining the level of PunchOut site 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.

## Designing the PunchOut site

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 PunchOut Site Requirements \[page 74\]](#)
- [Customer PunchOut Site Requirements \[page 74\]](#)
- [Learning About cXML and PunchOut \[page 75\]](#)

---

## Supplier PunchOut site 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 PunchOut site 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). HTTPS protects all parties in PunchOut sessions: your customer, Ariba Network, and your PunchOut site.

For more information, see [HTTPS Connections \[page 27\]](#).

## Learning about cXML and PunchOut

You need to document the transaction process flow into and out of your PunchOut site and identify which messages need to be coded. SAP 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

---

# Developing the PunchOut site

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

## PunchOut message flow

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

cXML messages for a PunchOut session include:

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.

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 **Checkout** 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 cart
2. Checkout

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.

## Related Information

[Purchase orders \[page 127\]](#)

---

## 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 site

The PunchOut testing phase ensures that your site is configured properly and that it effectively communicates with Ariba Buyer. Testing involves two steps: self-testing on Ariba Network, and subsequent testing with customers.

### Self-testing the PunchOut site 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.

#### **i** Note

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 topic in the **Product Documentation > For Administrators** section of the Learning Center.

## Testing the PunchOut site 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 include authentication, basic testing from Ariba Buyer.

## 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

There are a number of scenarios to test your PunchOut site, including testing PunchOut items, simulating a lost connection, testing multiple line items, non-catalog purchases, functionality of service items, and contract pricing,

- 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.

## Deploying the PunchOut site

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 SAP Ariba's PunchOut requirements, expediting the addition of new customers.

For more information, see [http://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 can 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

You can take steps to modify an existing e-commerce site to support PunchOut as rapidly as possible.

#### Procedure

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 PunchOut operation 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 session

The edit session PunchOut operation further describes a previously “created” 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, SAP Ariba recommends that you create a "PunchOut Deployment Guide" which explains your policies, capabilities, and processes, and give it to Ariba Buyer implementors to help them integrate with our site and adhere to your business policies.

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

- [PunchOut site connectivity overview \[page 83\]](#)
- [PunchOut site authentication and identification \[page 83\]](#)
- [PunchOut site required extrinsics \[page 84\]](#)
- [PunchOut site content requirements/specification \[page 84\]](#)
- [PunchOut site address information \[page 84\]](#)
- [PunchOut site accounting structure \[page 85\]](#)
- [PunchOut site commodity code description \[page 85\]](#)
- [PunchOut site transactions supported \[page 86\]](#)

## PunchOut site 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.

## PunchOut site 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.

---

## PunchOut site 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.

## PunchOut site 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.

## PunchOut site address information

Describe address format clearly and carefully 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
DeliverTo	1				
DeliverTo	2				
Street	1				
Street	2				
Street	3				
Street	4				

## PunchOut site 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 Pcard 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	
Reporting Center	Cost Center	Alpha / 5	ID	Y
Reporting Account	Account	Integer / 6	Account	N
Reporting Business	Company	Alpha / 4	ID	N

## PunchOut site commodity code description

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 \[page 377\]](#).

---

## PunchOut site 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.

### 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 \[page 154\]](#).

### 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 \[page 160\]](#).

### 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 , 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.

---

# PunchOut transactions

PunchOut transactions determine how your PunchOut site integrates with Ariba Network and procurement applications.

## In this section:

- [PunchOut index catalog \[page 88\]](#)
- [PunchOutSetupRequest document \[page 90\]](#)
- [PunchOutSetupResponse document \[page 96\]](#)
- [PunchOutOrderMessage document \[page 97\]](#)
- [PunchOut URL \[page 105\]](#)
- [URL vs. SelectedItem elements \[page 107\]](#)
- [Level 2 PunchOut \[page 107\]](#)
- [Level 2 PunchOut requirements \[page 110\]](#)
- [Example index catalogs \[page 111\]](#)
- [Shelf-level PunchOut \[page 115\]](#)
- [Uploading index catalogs \[page 115\]](#)
- [PunchOut session timeout \[page 116\]](#)
- [ASP examples \[page 116\]](#)

## 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.

## Related Information

- [SupplierID Element \[page 89\]](#)
- [URL Element \[page 89\]](#)
- [Classification Element \[page 89\]](#)
- [punchoutLevel Attribute \[page 89\]](#)
- [Example PunchOut Index File \[page 90\]](#)

---

## SupplierID element

The `SupplierID` 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 <http://www.dnb.com>

## URL element

In previous releases of Ariba Network, the `URL` element identified the supplier's PunchOut website. Now, the URL comes from the supplier's cXML profile or from the supplier's Ariba Network account.

The `URL` element must still be included, but it is no longer used by Ariba Network.

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

For more information, see [PunchOut URL \[page 105\]](#).

## 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 \[page 377\]](#).

## 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	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

Value	Use	Example
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 \[page 107\]](#).

## Example PunchOut index file

When you create your own PunchOut Level 2 index files, create them in cXML format to keep file size low. An example PunchOut file is provided as guidance.

```
<?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.

## Related Information

[Credential Elements \[page 91\]](#)  
[UserAgent Element \[page 92\]](#)  
[PunchOutSetupRequest operation Attribute \[page 92\]](#)  
[BuyerCookie Element \[page 93\]](#)  
[Extrinsic Element \[page 93\]](#)  
[BrowserFormPost Element \[page 94\]](#)  
[SupplierSetup Element \[page 94\]](#)  
[SelectedItem Element \[page 94\]](#)  
[Example PunchOutSetupRequest Document \[page 94\]](#)

## 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).

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

### To credential

This element identifies the supplier (the destination of the `PunchOutSetupRequest`.)

```
<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 \[page 39\]](#).

## 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

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

```
<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 \[page 98\]](#).



---

## 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

The optional `Extrinsic` 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.

### Note

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

The `BrowserFormPost` element contains the URL where the PunchOut site returns the `PunchOutOrderMessage` at the end of the PunchOut session.

```
<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. 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. PunchOut sites must still accept it until all Ariba Buyer installations recognize and send the `SelectedItem` element. For more information, see [PunchOut URL \[page 105\]](#).

## SelectedItem element

The `SelectedItem` 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.

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

## Example PunchOutSetupRequest document

The example `PunchOutSetupRequest` document demonstrates how a request 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`.

## Related Information

[Overview of the PunchOutSetupResponse Documents \[page 96\]](#)

[Status Element \[page 96\]](#)

[StartPage URL Element \[page 97\]](#)

[Example PunchOutSetupResponse Document \[page 97\]](#)

## Overview of the PunchOutSetupResponse documents

The `PunchOutSetupResponse` document indicates whether PunchOut initiation was successful, and 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

The `Status` 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 example `PunchOutSetupResponse` document demonstrates a successful request operation.

```
<?xml version="1.0"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML xml:lang="en-US" payloadID="933694607739"
  timestamp="2017-03-14T12:59:09-07:00">
  <Response>
    <Status code="200" text="success"></Status>
    <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.

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

Suppliers send `PunchOutOrderMessage` documents directly to the procurement application, not to the Ariba Network. Procurement applications do not interpret the `Sender` element in these documents, so suppliers should duplicate the information from the `From` element in the `Sender` element.

### Example

```
<From>
  <Credential domain="NetworkID">
    <Identity>AN01012345678</Identity>
  </Credential>
</From>
<To>
  <Credential domain="NetworkID">
    <Identity>AN01000002792</Identity>
  </Credential>
```

```
</To>
<Sender>
  <Credential domain="NetworkID">
    <Identity>AN01012345678</Identity>
  </Credential>
  <UserAgent>Our PunchOut Site V4.2</UserAgent>
</Sender>
```

Suppliers should not send a shared secret, because that value is confidential between each supplier and the Ariba Network.

## Related Information

[operationAllowed Attribute \[page 98\]](#)

[cxml-base64 and cxml-urlencoded \[page 98\]](#)

[BuyerCookie Element \[page 99\]](#)

[Total Element \[page 99\]](#)

[ItemIn Element \[page 100\]](#)

[ItemID Element \[page 100\]](#)

[ItemDetail Element \[page 101\]](#)

[Example PunchOutOrderMessage Document \[page 103\]](#)

## operationAllowed attribute

The `operationAllowed` 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

The `cxml-base64` and `cxml-urlencoded` attributes of the HTML `input` element identify 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.

## 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

The `BuyerCookie` 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 the cookie 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

The `Total` 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 Format \[page 17\]](#).

## ItemIn element

The `ItemIn` element passes PunchOut item details back to the requisition.

### Example

```
<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 \[page 139\]](#).

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

## ItemID element

The `ItemID` 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>
```

### **i** 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

The `ItemDetail` 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

The `UnitPrice` 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

The `Description` element contains the item description.

## UnitOfMeasure element

The `UnitOfMeasure` element contains the item Unit of Measure code as defined by the United Nations UOM standard.

For more information, see [Recommended Coding Systems \[page 377\]](#).

## PriceBasisQuantity element

The `PriceBasisQuantity` 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>
```

## Classification element

The `Classification` element contains the UNSPSC commodity code. Ariba Buyer 8.0 and later and Procure-to-Pay recognize an optional UNSPSC version number.

### Examples

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

For more information, see [Recommended Coding Systems \[page 377\]](#).

## LeadTime element

Ariba Buyer 8.2 (cXML 1.2.011) and later and SAP Ariba Buying and Invoicing 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 `PunchOutOrderMessage` example illustrates how the document is hidden inside a `cxml-urlencoded` form field.

```
<input type="hidden" name="cxml-urlencoded" value=
"<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.014/cXML.dtd">
<cXML payloadID="958074737352&www.workchairs.com"
timestamp="2004-06-14T12:59:09-07:00">
  <Header>
    <From>
      <Credential domain="DUNS">
        <Identity>12345678</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN01000002792</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="www.workchairs.com">
        <Identity>PunchoutResponse</Identity>
      </Credential>
      <UserAgent>Our PunchOut Site V4.2</UserAgent>
    </Sender>
  </Header>
  <Message>
    <PunchOutOrderMessage>
      <BuyerCookie>1J3YVWU9QWMTB</BuyerCookie>
      <PunchOutOrderMessageHeader operationAllowed="edit">
```

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.

104 CUSTOMER

## PunchOut URL

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

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 [Profile Transaction \[page 63\]](#).

---

If suppliers do not specify a profile URL on Ariba Network, it uses the method described in [URL Specified on Ariba Network \[page 106\]](#) 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)

The URL specified in the Supplier's Index catalog 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.

### Example

The URL from the index catalog appears in the `SupplierSetup` element for the `PunchOutSetupRequest` document. The following examples show 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

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.

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 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 \[page 89\]](#).

For details on the different levels of PunchOut items (for example, store, aisle, shelf, and product), see [punchoutLevel Attribute \[page 89\]](#) and the section on the `punchoutLevel` attribute in Chapter 2 of the .

---

## 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 steps illustrate 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

Static catalogs, store-level PunchOut index catalogs, and a Level 2 PunchOut index catalogs have required and optional fields.

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

Level 2 PunchOut include CIF and cXML index content and site requirements.

### CIF and cXML index content requirements

CIF and cXML index content includes both required and optional fields.

#### 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

There are certain minimum site requirements for Level 2 PunchOut as well as other possible 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 \[page 340\]](#) 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. For the smallest file, create the index catalog in cXML format instead of CIF. Both example index catalogs are in cXML format.

### Store-level PunchOut index catalog

The example index catalog example below shows store-level PunchOut with one PunchOut item.

```
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

The index catalog example shows Level 2 PunchOut, with expanded product descriptions.

```

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

The example PunchOutSetupRequest document illustrates the results of a simple user request from a level 2 PunchOut Index Catalog.

If a user selects the first item from the [Level 2 PunchOut Index Catalog \[page 112\]](#) example, a PunchOutSetupRequest document is generated.

```
<?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 Level 2 PunchOut Index Catalog example illustrates an item-level request.

```
<?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 [About Catalog Upload Transaction \[page 337\]](#).

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 ASP examples can be used as a starting point to deploy a PunchOut site rapidly. 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""
```

### Related Information

[receivePunchoutSetupRequest.asp \[page 117\]](#)

[resolveXML.asp \[page 118\]](#)

[TcXMLFormatDTime.asp \[page 119\]](#)



## receivePunchoutSetupRequest.asp

The `receivePunchoutSetupRequest.asp` example receiving the `PunchOutSetupRequest` document and generating 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 &&quot;@&&quot;";&
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 &&quot;@&&quot;";&
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%>&operation=<%= operation
%>&buyerCookie=<%= buyerCookie%>&BrowserFormPost=<%= BrowserFormPost%></URL></
StartPage></PunchOutSetupResponse></Response></cXML> <%end if%> </p>
```

## resolveXML.asp

The resolveXML.asp example 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 olddtdvalue
Dim newdtdvalue
olddtdvalue = ""cXML.dtd""
newdtdvalue = ""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,olddtdvalue,newdtdvalue)
    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

The TCcXMLFormatFDTime 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.
'
'Output:   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>

```

---

# Purchase requisitions

A purchase requisition is the first step in a procurement process.

A purchase requisition is a request to purchase an item (or multiple items). Each requisition is assigned a unique ID (such as PR2394) so you can track it as it moves through the purchasing process. A requisition can consist of multiple line items.

You send requisition documents in cXML format to the Ariba Procurement Solution through Ariba Network. After the requisition is processed into an order, the purchase order is sent to the appropriate supplier via Ariba Network.

A requisition can contain items from any of the following sources:

- The requestor's company catalog
- A supplier's catalog (also known as a PunchOut catalog)
- Non-catalog items (from another source)

## In this section:

[Importing requisitions \[page 121\]](#)

## Importing requisitions

Before you can import requisitions using cXML Posts, ensure that your site is configured to use XML for data integration. The cXML channel applies only to Ariba Network and not from the ERP to Ariba Procurement Solutions.

The following four extrinsic fields are required in the `PurchaseRequisitionRequest` cXML file:

- `AccountCategory`
- `BuyerPartNumber`
- `Facility`
- `Need-by Date`

## In this section:

[Importing a new requisition \[page 122\]](#)

[Updating a requisition \[page 124\]](#)

[Canceling a requisition \[page 125\]](#)

## Importing a new requisition

The following example imports a new requisition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.026/cXML.dtd">
<cXML payloadID="req00001"
  timestamp="2016-05-26T00:00:00-08:00" xml:lang="en-US">
  <Header>
    <From>
      <!-- Ariba Network buyer account -->
      <Credential domain="NetworkID">
        <Identity>AN71000002012</Identity>
      </Credential>
      <Credential domain="EndPointID">
        <Identity>ERP</Identity>
      </Credential>
    </From>
    <To>
      <!-- Ariba Network buyer account -->
      <Credential domain="NetworkID">
        <Identity>AN71000002012</Identity>
      </Credential>
      <Credential domain="EndPointID">
        <Identity>ERP</Identity>
      </Credential>
    </To>
    <Sender>
      <!-- This document has passed from the ERP
      to the Ariba Procurement Solution. -->
      <Credential domain="NetworkID">
        <Identity>AN71000002012</Identity>
      </Credential>
      <Credential domain="EndPointID">
        <Identity>ERP</Identity>
      </Credential>
      <SharedSecret>welcome3a</SharedSecret>
    </Sender>
    <UserAgent>Ariba.com Network V1.0</UserAgent>
  </Header>
  <Request>
    <PurchaseRequisitionRequest>
      <PurchaseRequisition>
        <PurchaseRequisitionHeader
          requisitionID="PR123"
          requisitionDate="2016-05-26T00:00:00-08:00"
          type="new">
          <ShipTo>
            <Address addressID="3000">
              <Name xml:lang="en">New York</Name>
              <PostalAddress>
                <DeliverTo>Joe Smith</DeliverTo>
                <Street>691 Random Ave</Street>
                <City>New York</City>
                <State>NY</State>
                <PostalCode>10001</PostalCode>
                <Country isoCountryCode="US">USA</Country>
              </PostalAddress>
            </Address>
          </ShipTo>
          <BillTo>
            <Address addressID="US006">
              <Name xml:lang="en">New York</Name>
              <PostalAddress>
                <Street>691 Random Ave</Street>
                <City>New York</City>
                <State>NY</State>
```

```

        <PostalCode>10001</PostalCode>
        <Country isoCountryCode="US">USA</Country>
    </PostalAddress>
</Address>
</BillTo>
<Contact role="preparer">
    <Name xml:lang="en-US">Jane Doe</Name>
    <PostalAddress>
        <Street>123 Anystreet</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>94089</PostalCode>
        <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
    <Email>cnoll@ariba.com</Email>
</Contact>
<Contact role="requester">
    <Name xml:lang="en-US">Jane Doe</Name>
    <PostalAddress>
        <Street>123 Anystreet</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>94089</PostalCode>
        <Country isoCountryCode="US">United States</Country>
    </PostalAddress>
    <Email>cnoll@ariba.com</Email>
</Contact>
</PurchaseRequisitionHeader>
<ItemIn quantity="10.000" lineNumber="00001">
    <ItemID>
        <SupplierPartID>MON923 6</SupplierPartID>
    </ItemID>
    <ItemDetail>
        <UnitPrice>
            <Money currency="USD">100.00</Money>
        </UnitPrice>
        <Description xml:lang="en">Optimax-V Monitor
            Cable DB9M/DB23F </Description>
        <UnitOfMeasure>EA</UnitOfMeasure>
        <Classification domain="UNSPSC">43211800</Classification>
        <Extrinsic name="AccountCategory">K</Extrinsic>
        <Extrinsic name="PurchaseOrg">3000</Extrinsic>
        <Extrinsic name="PurchaseGroup">100</Extrinsic>
        <Extrinsic name="BuyerPartNumber">SSP16446-cXML</Extrinsic>
        <Extrinsic name="Facility">Bangalore</Extrinsic>
        <Extrinsic name="Need-by Date">2016-06-10T00:00:00-08:00</Extrinsic>
    </ItemDetail>
    <SupplierList>
        <Supplier>
            <Name xml:lang="en">JCN Technologies</Name>
            <SupplierID domain="NetworkID">AN70000000004</SupplierID>
        </Supplier>
    </SupplierList>
    <Distribution>
        <Accounting name="Default">
            <AccountingSegment id="100">
                <Name xml:lang="en">Percentage</Name>
                <Description xml:lang="en">Percentage</Description>
            </AccountingSegment>
            <AccountingSegment id="02">
                <Name xml:lang="en">Company</Name>
                <Description xml:lang="en">ID</Description>
            </AccountingSegment>
            <AccountingSegment id="5000">
                <Name xml:lang="en">CostCenter</Name>
                <Description xml:lang="en">ID</Description>
            </AccountingSegment>
        </AccountingSegment id="US002">

```

```

        <Name xml:lang="en">BusinessUnit</Name>
        <Description xml:lang="en">ID</Description>
    </AccountingSegment>
    <AccountingSegment id="8100">
        <Name xml:lang="en">Account</Name>
        <Description xml:lang="en">ID</Description>
    </AccountingSegment>
    <AccountingSegment id="5009">
        <Name xml:lang="en">SubAccount</Name>
        <Description xml:lang="en">ID</Description>
    </AccountingSegment>
</Accounting>
<Charge>
    <Money currency="USD">20000.00</Money>
</Charge>
</Distribution>
</ItemIn>
</PurchaseRequisition>
</PurchaseRequisitionRequest>
</Request>
</cXML>

```

## Updating a requisition

The following example updates an existing requisition:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.026/cXML.dtd">
<cXML payloadID="req00002"
    timestamp="2016-05-27T00:00:00-08:00" xml:lang="en-US">
    <Header>
        <From>
            <!-- Ariba Network buyer account -->
            <Credential domain="NetworkID">
                <Identity>AN71000002012</Identity>
            </Credential>
            <Credential domain="EndPointID">
                <Identity>ERP</Identity>
            </Credential>
        </From>
        <To>
            <!-- Ariba Network buyer account -->
            <Credential domain="NetworkID">
                <Identity>AN71000002012</Identity>
            </Credential>
            <Credential domain="EndPointID">
                <Identity>ERP</Identity>
            </Credential>
        </To>
        <Sender>
            <!-- This document has passed from the ERP
            to the Ariba Procurement Solution. -->
            <Credential domain="NetworkID">
                <Identity>AN71000002012</Identity>
            </Credential>
            <Credential domain="EndPointID">
                <Identity>ERP</Identity>
                <SharedSecret>welcome3a</SharedSecret>
            </Credential>
            <UserAgent>Ariba.com Network V1.0</UserAgent>
        </Sender>
    </Header>
    <Request>

```



```

<PurchaseRequisitionRequest>
  <PurchaseRequisition>
    <PurchaseRequisitionHeader
      requisitionID="PR123"
      requisitionDate="2016-05-26T00:00:00-08:00"
      type="update">
        ...
    </PurchaseRequisitionHeader>
    <ItemIn>
      <ItemID>
        <SupplierPartID>MON923 6</SupplierPartID>
      </ItemID>
      <ItemDetail>
        <UnitPrice>
          <Money currency="USD">100.00</Money>
        </UnitPrice>
        <Description xml:lang="en">Optimax-V Monitor
          Cable DB9M/DB23F </Description>
        <UnitOfMeasure>EA</UnitOfMeasure>
        <Classification domain="UNSPSC">43211800</Classification>
        <Extrinsic name="AccountCategory">K</Extrinsic>
        <Extrinsic name="PurchaseOrg">3000</Extrinsic>
        <Extrinsic name="PurchaseGroup">100</Extrinsic>
        <Extrinsic name="BuyerPartNumber">SSP16446-cXML</Extrinsic>
        <Extrinsic name="Facility">Bangalore</Extrinsic>
        <Extrinsic name="Need-by Date">2016-06-10T00:00:00-08:00</Extrinsic>
      </ItemDetail>
    </ItemIn>
    ...
  </PurchaseRequisition>
</PurchaseRequisitionRequest>
</Request>
</cXML>

```

## Canceling a requisition

The following example cancels an existing requisition:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.026/cXML.dtd">
<cXML payloadID="req00002"
  timestamp="2016-05-27T00:00:00-08:00" xml:lang="en-US">
  <Header>
    <From>
      <!-- Ariba Network buyer account -->
      <Credential domain="NetworkID">
        <Identity>AN71000002012</Identity>
      </Credential>
      <Credential domain="EndPointID">
        <Identity>ERP</Identity>
      </Credential>
    </From>
    <To>
      <!-- Ariba Network buyer account -->
      <Credential domain="NetworkID">
        <Identity>AN71000002012</Identity>
      </Credential>
      <Credential domain="EndPointID">
        <Identity>ERP</Identity>
      </Credential>
    </To>
    <Sender>
      <!-- This document has passed from the ERP

```

```

to the Ariba Procurement Solution. -->
<Credential domain="NetworkID">
  <Identity>AN71000002012</Identity>
</Credential>
<Credential domain="EndPointID">
  <Identity>ERP</Identity>
  <SharedSecret>welcome3a</SharedSecret>
</Credential>
<UserAgent>Ariba.com Network V1.0</UserAgent>
</Sender>
</Header>
<Request>
  <PurchaseRequisitionRequest>
    <PurchaseRequisition>
      <PurchaseRequisitionHeader
        requisitionID="PR123"
        requisitionDate="2016-05-26T00:00:00-08:00"
        type="delete">
        ...
      </PurchaseRequisitionHeader>
      <ItemIn>
        <ItemID>
          <SupplierPartID>MON923 6</SupplierPartID>
        </ItemID>
        <ItemDetail>
          <UnitPrice>
            <Money currency="USD">100.00</Money>
          </UnitPrice>
          <Description xml:lang="en">Optimax-V Monitor
            Cable DB9M/DB23F </Description>
          <UnitOfMeasure>EA</UnitOfMeasure>
          <Classification domain="UNSPSC">43211800</Classification>
          <Extrinsic name="AccountCategory">K</Extrinsic>
          <Extrinsic name="PurchaseOrg">3000</Extrinsic>
          <Extrinsic name="PurchaseGroup">100</Extrinsic>
          <Extrinsic name="BuyerPartNumber">SSP16446-cXML</Extrinsic>
          <Extrinsic name="Facility">Bangalore</Extrinsic>
          <Extrinsic name="Need-by Date">2016-06-10T00:00:00-08:00</Extrinsic>
        </ItemDetail>
      </ItemIn>
      ...
    </PurchaseRequisition>
  </PurchaseRequisitionRequest>
</Request>
</cXML>

```

---

# Purchase orders

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.

## In this section:

- [OrderRequestHeader element \[page 127\]](#)
- [Masking values in blanket purchase orders and service purchase orders \[page 147\]](#)
- [Cancel orders and change orders \[page 154\]](#)
- [Change orders \[page 156\]](#)
- [Purchase order implementation hints and limitations \[page 158\]](#)
- [Schedule agreement releases \[page 173\]](#)
- [Planned and unplanned service line items \[page 176\]](#)
- [Response documents \[page 190\]](#)

## OrderRequestHeader element

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

### orderID attribute

The `orderID` attribute is a user-visible number generated by Ariba Buyer unique to the buying organization.

### orderDate attribute

The `orderDate` 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`.

### Example

```
<OrderRequest>
```

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

## orderType attribute

The `orderType` 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

The `addressID` attribute identifies the buying organization's shipping or billing address defined in the `ShipTo` and `BillTo` elements of `OrderRequest`. It 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, the `Name` element 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

The `DeliverTo` element on line one 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. Add it manually.

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

## DeliverTo element, line two

The `DeliverTo` element on line two 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.

### Note

Both `DeliverTo` lines in the `BillTo` element should be in the same format for consistency. This logic is not in the existing templates. Add it manually.

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

## Street element

The `Street` element is the street address of the `ShipTo` or `BillTo` location where ordered products are to be delivered and billed. Up to four `Street` elements are allowed to accommodate multi-line addresses.

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

---

## City element

The `City` element is the city where ordered products are to be shipped or billed.

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

## State element

The `State` element is a two-letter state, province, or territory code for the location where the goods are to be shipped and billed.

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

For more information, see [Country, State, and PostalCode Elements \[page 21\]](#).

## PostalCode element

The `PostalCode` 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.

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

For more information, see [Country, State, and PostalCode Elements \[page 21\]](#).

## CarrierIdentifier element

The `CarrierIdentifier` element contains the carrier name of the shipment.

```
<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>
  </Address>
</ShipTo>
```

## TransportInformation element

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

### Example

```
<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>
    <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>
  </ShipTo>
</OrderRequestHeader>
```

---

## Country isoCountryCode

The `isoCountryCode` attribute is the country code from the ISO 3166 standard. The content for `Country` is a human-readable or printable name.

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

For more information, see [Country, State, and PostalCode Elements \[page 21\]](#).

## TelephoneNumber element

The `TelephoneNumber` 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.

United States example:

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

London example:

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

For more information, see [Recommended Coding Systems \[page 377\]](#).

## Fax element

The `Fax` 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>
  </TelephoneNumber>
</Fax>
```

## Email element

The `Email` 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

The `TermsOfDelivery` 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.

### 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 \[page 211\]](#).

The `TermsofDelivery` element contains the following elements:

Element	Description
<code>TermsOfDeliveryCode</code>	Standard delivery terms.

Element	Description
ShippingPaymentMethod	<p>Denotes the mode of shipping payment.</p> <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>
TransportTerms	<p>The terms of transportation:</p> <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.

## Example

```

<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

The URL element is the ship to URL. Providing a value for this element is optional.

### Example

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

## Tax element

This element holds the details of the taxes on the purchase order. This element is present if the buying organization computes tax. When appearing within the `OrderRequestHeader` element, `Tax` describes the total tax for an order.

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

`Tax` elements at the line item level are nested within the `ItemOut` element and describe the taxes applied on the purchase order line items. Purchase orders generated by SAP Ariba Buying solutions have taxes only at the line level. In the example below, the tax code `Tax Code1` has been applied on the line item. The total tax amount for the tax code USD 10.96. Out of this, USD 8 is the amount for tax type `VAT` while USD 2.96 is the amount for tax type `Cess`.

```
<ItemOut>
:
:
  <Tax>
    <Money currency="USD">10.96</Money>
    <Description xml:lang="en">Tax Code1</Description>
    <TaxDetail category="VAT">
      <TaxAmount>
        <Money currency="USD">8.00</Money>
      </TaxAmount>
      <Description xml:lang="en">VAT Description</Description>
    </TaxDetail>
    <TaxDetail category="Cess">
      <TaxAmount>
        <Money currency="USD">2.96</Money>
      </TaxAmount>
      <Description xml:lang="en">Cess Description</Description>
    </TaxDetail>
  </Tax>
</ItemOut>
```

The `Money` element in `Tax` element is used to capture totals for all tax amounts and is not used for invoice display.

## Withholding Taxes

Two extrinsic elements can be added to the `Tax` element:

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

The example below shows 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. The use of negative rates ensures backward compatibility of existing cXML applications.

```
<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 contains details of the allowances and charges applicable at the header-level and line-item level.

This element provides details of:

- Any additional cost or deduction that alters the original price or shipping price of the item
- One or more `Modification` elements

The `Modification` element has an optional attribute `level`, which can have an integer value. It is used for displaying on which level a certain modification is applied. For more information, see [level Attribute \[page 266\]](#).

You can add the `Modifications` element to the `Shipping` element. Line-item level modifications can be added either at the line level, or at unit price level depending on buyer configurations. It has the following elements:

Element	Description
<code>OriginalPrice</code>	<p>The original price of the item. The allowances and charges are applied on the original price.</p> <p>This element has the <code>Money</code> element. It also contains an optional <code>type</code> attribute.</p> <p>For example, the <code>type</code> value can be the <code>MSRP</code>, <code>ListPrice</code>, <code>Actual</code>, <code>AverageSellingPrice</code>, <code>CalculationGross</code>, <code>BaseCharge</code>, <code>AverageWholesalePrice</code>, <code>ExportPrice</code>, <code>AlternatePrice</code>, <code>ContractPrice</code>, and so on.</p>
<code>AdditionalDeduction</code>	<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>• <code>DeductionAmount</code>: This element has the <code>Money</code> element.</li> <li>• <code>DeductionPercent</code>: This has a <code>percent</code> attribute.</li> <li>• <code>DeductedPrice</code>: This element has the <code>Money</code> element. It contains the final price of the item. This price overrides the price of the item.</li> </ul> <p>The <code>AdditionalDeduction</code> element has an optional <code>type</code> attribute. It contains details on the type of deductions available for the item.</p>
<code>AdditionalCost</code>	<p>The details of the additional charges applied on an item.</p> <p>It contains the following elements:</p> <ul style="list-style-type: none"> <li>• <code>Money</code>: Enter the money value in the <code>value</code> attribute. This is a mandatory attribute.</li> <li>• <code>Percentage</code>: Enter the percentage value in the <code>percent</code> attribute. This is a mandatory attribute.</li> </ul>
<code>ModificationDetail</code>	<p>The details of any information for the <code>AdditionalDeduction</code> or <code>AdditionalCost</code> element.</p> <p>This <code>ModificationDetail</code> element has the following elements:</p> <ul style="list-style-type: none"> <li>• <code>Description</code></li> <li>• <code>Extrinsic</code></li> <li>• <code>Code</code></li> </ul> <p>This element has the following attributes:</p> <ul style="list-style-type: none"> <li>• <code>name</code>: This is a mandatory attribute.</li> <li>• <code>startDate</code>: This is an optional attribute.</li> <li>• <code>endDate</code>: This is an optional attribute.</li> </ul>
<code>Tax</code>	<p>This tax is for the allowances and charges.</p> <div> <p><b>i Note</b></p> <p>The <code>OriginalPrice</code> element is an optional element when added as part of the <code>Modification</code> element.</p> </div>

## Example For Header-Level Modifications

```
<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>
</OrderRequestHeader>
```

## Example for Line-Item Level Modifications

In the example below, a discount of USD 2.00 and an insurance charge of USD 3.00 has been applied on a line item having amount USD 14.95.

```
<UnitPrice>
  <Money alternateCurrency="" alternateAmount="" currency="USD">14.95</Money>
  <Modifications>
    <Modification>
      <AdditionalDeduction>
        <DeductionAmount>
          <Money alternateCurrency="" alternateAmount=""
currency="USD">2.00</Money>
        </DeductionAmount>
      </AdditionalDeduction>
      <ModificationDetail name="Discount">
        <Description xml:lang="en-US">Discount</Description>
      </ModificationDetail>
    </Modification>
    <Modification>
      <AdditionalCost>
        <Money alternateCurrency="" alternateAmount="" currency="USD">3.00</
Money>
      </AdditionalCost>
      <ModificationDetail name="Insurance">
        <Description xml:lang="en-US">Insurance Charges</Description>
      </ModificationDetail>
    </Modification>
  </Modifications>
</UnitPrice>
```

## ItemOut element

ItemOut elements describe individually ordered items.

The ItemOut element can contain a `requiresServiceEntry="yes"` attribute, which specifies that the supplier must create a service sheet for the line item.

The ItemOut element can also contain the `ItemType` and `parentLineNumber` attributes to specify if the line item is an item group having child line items 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. ItemOut elements with either `ItemType` can require a service entry sheet. Buyers can group child line items under an item group.

### 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>
  </ItemDetail>
</ItemOut>
```

```

        </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.

The `Distribution` element 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.

The `Distribution` element can contain `Accounting`, `AccountingSegment`, and `Segment` 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.

### Example

```

<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 the 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 the 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 dependent on position in the accounting structure, Buying organizations can provide only the ones they wish to represent.

### **i** 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.

## ControlKeys element

The `ControlKeys` element in the cXML structure allows you to override default business rules for order confirmations, ship notices, service sheets, and invoices. `ControlKeys` has the following elements:

Element	Description
<code>OCInstruction</code>	Indicates whether an order confirmation is allowed for this order or line item, regardless of the default business rules configured in the Ariba Network.
<code>ASNInstruction</code>	Indicates whether a ship notice is allowed for this order or line item, regardless of the default business rules configured in the Ariba Network.
<code>SESInstruction</code>	Indicates whether a service sheet is allowed for this order or line item, regardless of the default business rules configured in the Ariba Network.
<code>InvoiceInstruction</code>	Indicates whether an invoice is allowed for this order or line item, regardless of the default business rules configured in the Ariba Network.

For example, you can use the `InvoiceInstruction` child element to differentiate between regular invoices and ERS invoices. In the following example, the `InvoiceInstruction@value` attribute is set to "isERS", which disables the creation of invoices against a line item flagged for ERS on the purchase order.

## Example

```
<ItemOut lineNumber="1" quantity="2" requestedDeliveryDate="2008-09-21">
  <ItemID>
    <SupplierPartID>AX4518</SupplierPartID>
    <SupplierPartAuxiliaryID>AXSPA001</SupplierPartAuxiliaryID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">31.20</Money>
    </UnitPrice>
    <Description xml:lang="en">BULLNOSE SHELVES 4 PK</Description>
    <UnitOfMeasure>PK</UnitOfMeasure>
    <Classification domain="SPSC">foo</Classification>
    <ManufacturerPartID>AX4518</ManufacturerPartID>
    <ManufacturerName>20008496</ManufacturerName>
  </ItemDetail>
  <Distribution>
    <Accounting name="DistributionCharge">
      <Segment description="Entertainment" id="23456" type="G/L Account"/>
      <Segment description="Western Region Sales" id="2323" type="Cost Center"/>
    </Accounting>
    <Charge>
      <Money currency="USD">187.20</Money>
    </Charge>
  </Distribution>
  <ControlKeys> <InvoiceInstruction value="isERS"/> </ControlKeys>
</ItemOut>
```

## AccountingSegment element

The `AccountingSegment` element identifies the accounting segment relative to others in the `Accounting` 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.

## Example

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

## Segment element (deprecated)

The `Segment` element identifies the accounting segment relative to others in the `Accounting` element. `Segment` is deprecated, replaced by the `AccountingSegment` element in cXML 1.2.005, but supplier must be able to process `Segment` because it is still generated by Ariba Buyer 6, 7, and 8.

`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). `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. The values can also include accounting information for Funds Management account assignment fields. Examples of Funds Management account assignment fields are Fund, Budget Period, Funds Center, Commitment Item, Earmarked Funds Document, Functional Area, Grant, and FM Area.

### Example

```

<Distribution>
  <Accounting name="DistributionCharge">
    <Segment description="Travel" id="456" type="G/L Account"/>
    <Segment description="European Projects" id="23" type="Cost Center"/>
    <Segment description="Fund001" id="301" type="Fund" />
    <Segment description="BP2015" id="BP1" type="Budget Period" />
    <Segment description="FC1000" id="1200" type="Funds Center" />
  </Accounting>
  <Charge>
    <Money currency="USD">4688.00</Money>
  </Charge>
</Distribution>

```

In the example `Travel` describes the G/L Account charge, and corresponds to the 456 account. This example also includes Funds Management accounting information for the Fund, Budget Period, and Funds Center fields.

# Accounting fields in SAP Ariba Invoice Management

Purchase orders sent to SAP Ariba Invoice Management have expected cXML mappings for accounting fields split fields.

## Example

```
<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>
```

## Receiving types in SAP Ariba Invoice Management

Many ERP systems expect a match between an invoice and the associated orders for items that do not require a receipt. These ERP systems typically indicate if a receipt is required 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.

### Example

```
<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>
  . . .
```

If `ReceivingType` is any other value than 4 or 10, or if `ReceivingType` is not present in the `ItemDetail` element, the invoice reconciliation includes matching of receipts to invoices and orders. Supplier see the receiving type in Ariba Network when reviewing order details.

## ReceivingType integration with external systems

`ReceivingType` integration with an external ERP system depends on how the external system handles receipt requirements. For example, some ERP systems only indicate at the header level if a receipt is required. You 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 unplanned service order items.

### **i** Note

The Ariba Network does not support service lines in blanket purchase orders. Service lines are only supported in service purchase orders.

## BlanketItemDetail in blanket purchase orders

The `BlanketItemDetail` element can appear in three different kinds of blanket purchase orders: supplier, commodity, and item.

- **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 the creation of a BPO.
- `Ariba.invoicingAllowed`: A Boolean value that specifies 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 that specifies if invoices can be created by supplier punch in.

### **i** Note

The `BlanketItemDetail` element can also contain the quantity-based pricing for a line item in a blanket purchase order.

## Related Information

[PriceBasisQuantity element \[page 150\]](#)

## BlanketItemDetail in service purchase orders

BlanketItemDetail describes line items in a service purchase order that are unplanned and do not have any specifics. When suppliers fulfill the order, they add items to the service sheet to describe how they fulfilled the order.

For unplanned service lines, the BlanketItemDetail element must enclose a MaxAmount element that defines the hidden maximum amount that the supplier cannot exceed when creating service sheets.

### Example

```
<ItemOut lineNumber="1" quantity="1" requestedDeliveryDate="2013-09-1"
requiresServiceEntry="yes">
  <ItemID>
    <SupplierPartID>LW4001</SupplierPartID><SupplierPartAuxiliaryID></
SupplierPartAuxiliaryID>
  </ItemID>
  <BlanketItemDetail>
    <Description xml:lang="en">Site preparation of parcel no. 110 (3 acres)</
Description>
    <MaxAmount>
      <Money currency="USD">70000</Money>
    </MaxAmount>
    <UnitPrice><Money currency="USD">50000</Money></UnitPrice>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="ascc">606501</Classification>
    <Extrinsic name="Construction"></Extrinsic>
    <Extrinsic name="ServicePeriod">    <Period
endDate="2013-09-01T23:59:59-00:00"
      startDate="2012-07-12T00:00:00-00:00"></Period>    </Extrinsic>
    </BlanketItemDetail>
  </ItemOut>
```

## Masking values in blanket purchase orders and service purchase orders

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

### Note

If a supplier creates an invoice item for a blanket purchase order item, the quantity and unit price values are editable for the invoice item.

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 or service 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 or service purchase order from its Ariba Network buyer account. When the blanket or service purchase order contains a masked value, the string “The purchase order contains masked values for the supplier.” appears. When the blanket or service 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 or service purchase order. For more information, see the *Ariba Network Buyer Administration Guide*.

## Header-level and line-level masking

The `hideAmount` and `hideUnitPrice` extrinsics can be applied to mask values at the header and line levels.

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 or service 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.:

Extrinsic Enabled	Values Masked in a Blanket or Service Purchase Order					
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	-

## AmountBasedReceiving extrinsic

Line items that contain the `AmountBasedReceiving` extrinsic with the value “Yes” are considered amount-based line items. Amount-based line items can be used in purchase orders for services, and only need to specify the total amount that the buyer expects to spend.

When a supplier invoices an amount-based line item, the **Quantity**, **Unit**, and **Unit Price** values are not displayed. Suppliers can enter partial amounts to partially invoice amount-based line items.



## Example

```
<ItemDetail>
  <UnitPrice>
    <Money alternateCurrency="" alternateAmount="" currency="USD">200</Money>
  </UnitPrice>
  <Description xml:lang="en">IT Services</Description>
  <UnitOfMeasure>EA</UnitOfMeasure>
  <Classification domain="unspsc">76</Classification>
  <LeadTime>0</LeadTime>
  <Extrinsic name="AmountBasedReceiving">Yes</Extrinsic>
</ItemDetail>
```

## ContingencyReceiving extrinsic

Line items that contain the extrinsic `ContingencyReceiving` with the value "Yes" are considered contingency line items. Contingency line items can be used in purchase orders for services and this line type describes a service that is not essential to perform an purchase order. The value of the contingency line is included in the total value of the Bill of Services (BOS) only if the contingency line item is enabled while creating a service sheet or an invoice.

## Example

```
<ItemDetail>
- <UnitPrice>
  <Money currency="USD">50</Money>
</UnitPrice>
<Description xml:lang="en">bulldozer driver</Description>
<UnitOfMeasure>HUR</UnitOfMeasure>
<Classification domain="SPSC">foo</Classification>
<ManufacturerPartID>AX4518</ManufacturerPartID>
<ManufacturerName>20008496</ManufacturerName>
<Extrinsic name="ContingencyReceiving">Yes</Extrinsic>
</ItemDetail>
```

## InformationReceiving extrinsic

Line items that contain the `InformationReceiving` extrinsic with the value "Yes" are considered information service lines. Information service lines are used only to store additional information; service sheets and invoices cannot be created against information service lines.

## Example

```
<ItemDetail>
```

```
-<UnitPrice>
<Money currency="USD">3000</Money>
</UnitPrice>
<Description xml:lang="en">bulldozer minia</Description>
<UnitOfMeasure>PK</UnitOfMeasure>
<Classification domain="SPSC">foo</Classification>
<ManufacturerPartID>AX4518</ManufacturerPartID>
<ManufacturerName>20008496</ManufacturerName>
<Extrinsic name="InformationReceiving">Yes</Extrinsic>
</ItemDetail>
```

## Money currency attribute

The currency attribute is an ISO 4217 standard 3-letter currency code that indicates how to interpret the monetary value expressed in the `Money` element.

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

For more information, see [Recommended Coding Systems \[page 377\]](#).

## Modifications Element

The Modification element contains details of the allowances and charges applicable for line items. The Modifications element is wrapped by an `ItemDetail` element.

## TermsOfDelivery element

The `TermsOfDelivery` element specifies the terms of delivery in a purchase order or ship notice.

## PriceBasisQuantity element

The `PriceBasisQuantity` element contains the quantity-based pricing for a line item. Quantity-based pricing allows the unit price of an item to be calculated on quantities greater 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

```
<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 unit price is for 2 Boxes</Description>
  </PriceBasisQuantity>

```

### **i** Note

This element can be used in blanket purchase orders, but not service purchase orders.

## Tolerances element

The `Tolerances` element, available to buyers on Ariba Network, allows buyers to specify maximum price limits for line items in a purchase order, and minimum and maximum quantity limits at purchase order header and line item level.

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 order confirmations, ship notices, or service sheets against the purchase order.

When a buyer sends a changed purchase order containing a line item quantity tolerance, the 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>• <code>Percentage</code>: The percentage for the quantity tolerance.</li> <li>• <code>Value</code>: The quantity number for the quantity tolerance.</li> </ul> <p>You can specify one of these elements in the <code>QuantityTolerance</code> element.</p>

Element	Description
PriceTolerance	<p>The price tolerance for a line item. This element has the following elements:</p> <ul style="list-style-type: none"> <li>Percentage: The percentage for the price tolerance.</li> <li>Money: The amount and currency for the price tolerance.</li> </ul> <p>You can specify one of these elements in the PriceTolerance element.</p>

Quantity tolerances must be enclosed in Lower and Upper elements to define minimum and maximum tolerances.

## Examples

```
<ItemOut>
  <Tolerances>
    <QuantityTolerance>
      <Percentage percent = "12"/>
    </QuantityTolerance>
    <PriceTolerance>
      <Money currency="USD">200</Money>
    </PriceTolerance>
  </Tolerances>
</ItemOut>
```

```
<ItemOut>
  <ControlKeys>
    <OCInstruction value="allowed">
      <Lower>
        <Tolerances>
          <QuantityTolerance>
            <!-- underdelivery of up to 5 units allowed -->
            <Value value="5" />
          </QuantityTolerance>
        </Tolerances>
      </Lower>
      <Upper>
        <Tolerances>
          <QuantityTolerance>
            <!-- up to 10% overdelivery allowed -->
            <Percentage percent="10" />
          </QuantityTolerance>
        </Tolerances>
      </Upper>
    </OCInstruction>
    <...>
  </ControlKeys>
</ItemOut>
```

## Additional References

- *Ariba Network Buyer Administration Guide*

## ScheduleLine element

The `ScheduleLine` element specifies information related to delivery schedules for a line item. A line item can have a maximum of 50 schedule lines.

`ScheduleLine` wraps a `UnitOfMeasure` element and has the following attributes:

Attribute	Description
<code>quantity</code>	Quantity of items to be shipped. This is a mandatory field.
<code>requestedDeliveryDate</code>	Date that the specified quantity is expected to be delivered This is a mandatory field.
<code>deliveryWindow</code> (optional)	Duration of time in which the quantity is expected to be delivered.
<code>lineNumber</code> (optional)	A line number can be added as a line identifier for a specific schedule line.

## Ariba Network currency code mapping

Use ISO 4217 three-letter (CodeA) currency codes to specify currency types in cXML. Ariba Network recognizes some formerly supported codes that are now obsolete, and converts them to their new codes.

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 them to USD.

### Note

Ariba Network does not perform currency code conversion for test accounts.

## Related Information

[Currency Codes \[page 378\]](#)

# 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.

## Cancel and change order requirements for buying organizations

Cancel and change orders use depends on the business processes of buying organizations. Organizations 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. Ariba Network has buyer-configurable rules, however, for allowing cancel and change orders for fully shipped or partially shipped purchase orders. Suppliers can view these rules.

Ariba Network allows buying organizations to change service purchase orders that are marked Partially Serviced, with the following limitations:

Purchase Order Status	Changes Allowed
New	Buyers can make any updates except changing a service line ( <code>requiresServiceEntry="yes"</code> ) to a non-service line or vice versa. Service lines must remain service lines and non-service lines must remain non-service lines in any changes to existing orders.
Partially Serviced	<ul style="list-style-type: none"><li>• Buyers cannot change service lines to non-service lines or vice versa at any time.</li><li>• Buyers can add both service and material goods lines.</li><li>• Buyers can edit or delete both service and material goods lines that have approved or unapproved service sheets created against them.</li></ul>

Buyers can update purchase order or line amounts on service orders for projects that go over initial budgets so that suppliers can continue to create service sheets against them until the project is completed.

Suppliers can create new service sheets against changed purchase orders. They can only invoice approved service lines if the buyer has not removed those lines from the underlying order.

---

## Cancel and change order requirements for 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 Orders \[page 202\]](#) and [Ship Notices and Cancel and Change Orders \[page 211\]](#).

## 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 "Obsoleted."

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`.

### Example

```
<?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
                .
                .
            </OrderRequestHeader>
        </OrderRequest>
    </Request>
</cXML>

```

For information about the interaction between these orders and confirmation and ship notices, see [Confirmations and Cancel Orders \[page 202\]](#) and [Ship Notices and Cancel and Change Orders \[page 211\]](#).

## 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 from Ariba Buyer 8.1 and later.



## Change orders in Ariba Buyer 8.0 and earlier

Ariba Buyer 8.0 and earlier denote a change order by appending -Vn 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 subsequent change order.

Ariba Buyer 7.0.6 fixed a minor problem in cXML change orders. For more information, see [Changes Introduced by Ariba Buyer 7.0.6 \[page 183\]](#).

## Change orders in Ariba Buyer 8.1 and later

Ariba Buyer 8.1 and later employ cXML 1.2009 with `OrderRequestHeader` attributes to insert order version information that enable better version tracking by receiving systems.

`OrderRequestHeader` attributes include:

- `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.

### 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"`).

---

# Purchase order implementation hints and limitations

When you implement your cXML purchase order applications be aware of recommendations and possible limitations.

## In this section:

- [Credential elements \[page 159\]](#)
- [Contact roles \[page 159\]](#)
- [Sold To Address and VAT ID \[page 159\]](#)
- [Path routing \[page 160\]](#)
- [Non-catalog items \[page 160\]](#)
- [User-entered information for non-catalog items \[page 160\]](#)
- [Enabling or disabling non-catalog orders \[page 161\]](#)
- [Detecting non-catalog orders \[page 161\]](#)
- [Breaking requisitions into multiple OrderRequests \[page 161\]](#)
- [Item groups and grouped items in service orders \[page 161\]](#)
- [Attachments \[page 162\]](#)
- [Attachments on Ariba Network \[page 162\]](#)
- [Attachment file names \[page 162\]](#)
- [AttachmentOnline extrinsic \[page 163\]](#)
- [Service sheets \[page 163\]](#)
- [PCards \[page 164\]](#)
- [Blanket purchase orders \[page 164\]](#)
- [Subcontracting orders \[page 164\]](#)
- [Incoterms on orders \[page 172\]](#)
- [AribaNetwork.PaymentTermsExplanation extrinsic \[page 173\]](#)

## Related Information

- [Maximum document size \[page 15\]](#)
- [Extrinsic for clickable links \[page 181\]](#)
- [Extrinsic for legacy purchase orders \[page 183\]](#)
- [Changes introduced by Ariba Buyer 7.0.6 \[page 183\]](#)

## Credential elements

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

For detailed information about the credentials used in `OrderRequest` documents, see [About cXML credentials \[page 35\]](#).

For information about credential limitations of older versions of Ariba Buyer, see [NetworkID and older versions of Ariba Buyer \[page 39\]](#).

## 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 provide 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
"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. As a best practice, Ariba recommends that buyers using PO Automation or Ariba Invoice Professional include the Sold To address and VAT ID in the purchase order cXML.

Including the Sold To address and VAT ID in purchase order cXML eliminates the need for the supplier to provide this information on a subsequent 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>
```

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.

---

## 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. Copy organizations 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. Copy organizations 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 rather than 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.

## User-entered information for non-catalog items

Non-catalog items must include specific user-entered information.

Non-catalog item information should include:

- Supplier Part ID (optional)
- Description
- Commodity Code
- Quantity
- Unit of Measure
- Unit Price

Suppliers must carefully validate this information since it does not come from their electronic catalogs.

## Enabling or disabling non-catalog orders

Buying organizations determine whether their users can order non-catalog items. Suppliers should contact their customers to determine if they allow 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 (Ariba 8.0 and later), or Supplier Part IDs that are blank or set to "not available".

Ariba Buyer does not use the `isAdHoc` flag 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 can then automatically process as many requisition items as possible, instead of having to manually process both catalog and non-catalog items.

## Item groups and grouped items in service orders

In purchase orders with multiple levels (item groups that contain grouped items), service lines can be either individual lines, item groups, or grouped items. Ariba Network does not support service lines under a non-service item group.

Service lines can be grouped under a service item group. For example:

Allowed:	Not Allowed:
<ul style="list-style-type: none"><li>• 1 Service Line Item A (item group)<ul style="list-style-type: none"><li>◦ 1.1 Goods Line Item A1 (grouped item)</li><li>◦ 1.2 Goods Line Item A2 (grouped item)</li></ul></li><li>• 2 Service Line Item B (item group)<ul style="list-style-type: none"><li>◦ 2.1 Goods Line Item B1 (grouped item)</li><li>◦ 2.2 Service Line Item B2 (grouped item)</li></ul></li><li>• 3 Service Line Item C (individual line)</li><li>• 4 Goods Line Item D (individual line)</li></ul>	<ul style="list-style-type: none"><li>• 1 Goods Line Item A (item group)<ul style="list-style-type: none"><li>◦ 1.1 Service Line Item A1 (grouped item)</li></ul></li></ul>

---

## Attachments

Requisitioners sometimes clarify purchase orders with 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.

The order attachment implementation on Ariba Network are as follows:

- 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.

### Example

```
<Extrinsic name="AttachmentOnline">  
  https://service.ariba.com/ad/orderDetail?poID=1234&amp;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.

#### Note

Ariba Network inserts escaped ampersand (&) characters in the URL. Before you can use the URL to log in and view the purchase order, replace "&amp;" with "&" in the URL.

Ariba Network adds this `Extrinsic` only for suppliers that have selected **Leave attachments online...** in their order routing configuration.

## Service sheets

Ariba Network requires suppliers to create one or more service sheet documents for any line in a purchase order with an `ItemOut` attribute that includes the `requiresServiceSheet= "yes"` attribute.

Service sheets can include both service and material goods lines. The supplier uses the service sheets to describe services as they are performed, and also uses them as a basis for invoices. See [About Service Sheets \[page 217\]](#).

## PCards

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.

## Blanket purchase orders

SAP Ariba Procurement solutions only support item-level Blanket Purchase Orders (BPOs) from external ERP systems.

When SAP Ariba Procurement 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, but 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.

### Note

Ariba Network does not support service lines (those that require a service sheet) in blanket purchase orders.

## Subcontracting orders

A subcontracting order is a purchase order that is sent from a buyer to a contract manufacturer to request the production and delivery of finished goods.

The subcontracting order carries not only item level and schedule line level information but also subcontracting component information. Subcontracting components are the raw materials that are used for manufacturing the finished goods specified at the item level.

### Subcontracting order

The following elements and attributes have been added to support contract manufacturing collaboration.

Table 1: Elements added for the subcontracting order

Element	Contained In	Description
<code>SubcontractingComponent*</code>	<code>ScheduleLine</code>	Contains a buyer's detailed information about a subcontracting component which is used to manufacture finished goods. For example, <code>SubcontractingComponent</code> could contain an ID, a description, a buyer's product ID, a quantity, or the date required.



Element	Contained In	Description
ComponentID	SubcontractingComponent	Contains a buyer's identifier for a subcontracting component within the procurement process.
Product	SubcontractingComponent	Contains product information for a subcontracting component, such as a buyer product ID, supplier product ID, standard product ID, or internal product ID.
ProductRevisionID	SubcontractingComponent	Contains an identifier for a specific change made to a component. This is a sequential number that is assigned when changes are made to an object.
Batch	SubcontractingComponent ItemOut	Contains batch information for material or goods produced in a single manufacturing run.
BuyerBatchID	Batch	Contains a buyer's identifier for the material or goods produced in a single manufacturing run.
PropertyValuation	Batch	Contains a value that can be assigned to characteristics, such as property values pertaining to currency amounts, quantities, or dates. <i>PropertyValuation</i> includes the property to be valued and the associated values.
PropertyReference	PropertyValuation	A reference to the property being valued.
ValueGroup	PropertyValuation	Contains a group of values for a property being valued.
ParentID	ValueGroup	An identifier of its parent.
PropertyValue	ValueGroup	Properties for valuating a property.

Table 2: Attributes added for the subcontracting order

Attribute	Contained In	Description
itemCategory	ItemOut	Defines how a component or material is procured. Possible values are subcontract, consignment, or thirdParty.
code	Characteristic	Code associated with a characteristic, such as a currency code or unit of measure.

## Component ship notice message

Table 3: Elements added for the component ship notice

Element	Contained In	Description
IdReference	ItemID	Contains a group of values pertaining to a property. IdReference was added to ItemID to refer to additional identifiers.
Batch	ShipNoticeItem	Contains batch information for material or goods produced in a single manufacturing run.

## Component inventory snapshot

Table 4: Elements added for the component inventory snapshot

Element	Contained In	Description
ProductActivityMessage	n/a	Transmits component inventory, consignment movement, and forecast information from the buyer's ERP system. The buyer-provided inventory summary view includes the components issued to the supplier. The information provided gives a snapshot of the component inventory and forecast situation at a certain point in time.
ProductActivityHeader	ProductActivityMessage	Contains information about of a single component inventory, the product forecast details, or a consignment movement for the product.
ProductActivityDetails	ProductActivityMessage	Contains the details about a product activity.
ItemID	ProductActivityDetails	Contains identifying information for the item.
Batch	ProductActivityMessage	Contains batch information for material or goods produced in a single manufacturing run.
Contact	ProductActivityDetails	The contact element contains, in the role attribute, the name of the plant at the point of origin, and the name of the plant at the destination.  Use the role attribute, and the IdReference and Description elements to pass plant information.
IdReference	Contact	You can pass plant information in IdReference by setting values for the domain and identifier attributes.

Element	Contained In	Description
Description	ProductActivityDetails	<p><b>i Note</b></p> <p>To use <code>IdReference</code> to pass plant information, set plant name in the description, as in the following example:</p> <pre>&lt;Description xml:lang="en"&gt;Sunnyvale - Plant1&lt;/Description&gt;</pre>
Inventory	ProductActivityDetails	Contains details about the component stock.
SubcontractingStockIn-TransferQuantity	Inventory	Contains information about quantity and unit of measure.

Table 5: Attributes added for the component inventory snapshot

Attribute	Contained In	Description
role	Contact	<p>When role is set to <code>locationFrom</code> or <code>locationTo</code> then plant information can be passed in <code>IDReference</code>.</p> <p>To use <code>Contact</code> for point of origin, set <code>role=locationFrom</code>. To use <code>Contact</code> for destination, set <code>role=locationTo</code>.</p>

## Component receipt message

Table 6: Attributes added for the component receipt message

Attribute	Contained In	Description
completedIndicator	ReceiptItem	Specifies whether a component ship notice item is considered closed. This would mean that no more component receipts are expected for this item. This attribute is a pass-through value and no validation is done against it in the backend.
lineNumber	ReceiptItemReference	In the scenario where the <code>OrderRequest</code> is blank, the line number refers to the <code>shipNoticeLineNumber</code> of the Component Ship Notice, that is, the ship notice referenced in the <code>ShipNoticeReference</code> .

## Component consumption (backflush)

Table 7: Elements added for the component consumption (backflush) message

Element	Contained In	Description
ComponentConsumption-Details	ShipNoticeItem	Contains component consumption details.

## Component consumption (real-time)

Table 8: Elements added for the component consumption (real-time) message

Element	Contained In	Description
ComponentConsumptionRequest	n/a	Data sent by a supplier to the buyer to report the consumption of components during the manufacturing of an ordered item.
ComponentConsumptionHeader	ComponentConsumptionRequest	Contains data pertinent to all portions of the component consumption.
ComponentConsumptionPortion	ComponentConsumptionRequest	Contains details of all component consumptions for a particular OrderRequest.
ComponentConsumptionItem	ComponentConsumptionPortion	Contains purchase order details for the components consumed.
ComponentConsumptionDetails	ComponentConsumptionItem	Contains details about component consumption.

## Subcontracting order example

The following excerpt from a subcontracting order shows an `ItemOut` element containing several `SubcontractingComponent` elements:

```
<ItemOut itemCategory="subcontract" requestedDeliveryDate="2008-09-21"
quantity="100" lineNumber="1">
  <ItemID>
    <SupplierPartID>AX4518-1</SupplierPartID>
    <SupplierPartAuxiliaryID>AXSPA001-1</SupplierPartAuxiliaryID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">31.20</Money>
    </UnitPrice>
    <Description xml:lang="en">Violin</Description>
    <UnitOfMeasure>PK</UnitOfMeasure>
    <Classification domain="SPSC">foo</Classification>
    <ManufacturerPartID>AX4518-1</ManufacturerPartID>
```

```

        <ManufacturerName>20008496-1</ManufacturerName>
    </ItemDetail>
    <Distribution>
        <Accounting name="DistributionCharge">
            <Segment type="G/L Account" id="23456" description="Entertainment"/>
            <Segment type="Cost Center" id="2323" description="Western Region
Sales"/>
        </Accounting>
        <Charge>
            <Money currency="USD">31.20</Money>
        </Charge>
    </Distribution>
    <ScheduleLine requestedDeliveryDate="2008-09-20T00:00:00-08:00" quantity="10"
lineNumber="1">
        <UnitOfMeasure>PK</UnitOfMeasure>
        <SubcontractingComponent quantity="1"
requirementDate="2013-08-01T14:37:31-07:00">
            <ComponentID>1</ComponentID>
            <UnitOfMeasure>BOX</UnitOfMeasure>
            <Description xml:lang="en">Fiddle Peg</Description>
            <Product>
                <SupplierPartID>SupplierPartID1</SupplierPartID>
                <SupplierPartAuxiliaryID>SupplierPartAuxiliaryID1</
SupplierPartAuxiliaryID>
                <BuyerPartID>BuyerPartID1</BuyerPartID>
                <IdReference domain="StandardID" identifier="1000"/>
                <IdReference domain="InternalID" identifier="2000"/>
            </Product>
            <ProductRevisionID>12345</ProductRevisionID>
            <Batch originCountryCode="US"
expirationDate="2014-06-10T14:37:31-07:00"
productionDate="2013-06-10T14:37:31-07:00">
                <BuyerBatchID>BuyerBatchID1</BuyerBatchID>
                <SupplierBatchID>SupplierBatchID1</SupplierBatchID>
                <PropertyValuation>
                    <PropertyReference>
                        <IdReference domain="ID" identifier="Image"/>
                    </PropertyReference>
                    <ValueGroup>
                        <PropertyValue name="BinarySpecification">
                            <Characteristic domain="BinaryObject"
value="AQIDBAUGBwgJAA==" />
                            <Characteristic domain="mimeCode" value="x-world/
x-3dmf"/>
                            <Characteristic domain="characterSetCode"
value="ASCII"/>
                            <Characteristic domain="format" value="netCDF"/>
                            <Characteristic domain="fileName" value="binaryData"/>
                            <Characteristic domain="uri" value="http://sap.com/
anyURI"/>
                        </PropertyValue>
                        <PropertyValue name="DateTimeSpecification">
                            <Characteristic domain="DateTime"
value="1999-05-31T13:20:00Z"/>
                            <Characteristic domain="timeZoneCode" value="PST"/>
                            <Characteristic domain="daylightSavingTimeIndicator"
value="true"/>
                        </PropertyValue>
                        <PropertyValue name="AmountSpecification">
                            <Characteristic domain="Amount" value="200" code="USD"/>
                            <Characteristic domain="LowerAmount" value="100"
code="EUR"/>
                            <Characteristic domain="UpperAmount" value="300"
code="EUR"/>
                        </PropertyValue>
                        <PropertyValue name="QuantitySpecification">
                            <Characteristic domain="Quantity" value="200"
code="BOX"/>

```

```

                                <Characteristic domain="LowerQuantity" value="100"
code="EA"/>
                                <Characteristic domain="UpperQuantity" value="300"
code="EA"/>
                                </PropertyValue>
                            </ValueGroup>
                        </PropertyValuation>
                    <PropertyValuation>
                        <PropertyReference>
                            <IdReference domain="ID" identifier="Price"/>
                        </PropertyReference>
                    </ValueGroup>
                        <PropertyValue name="DateTimeSpecification">
                            <Characteristic domain="DateTime"
value="1999-05-31T13:20:00Z"/>
                            <Characteristic domain="timeZoneCode" value="PST"/>
                            <Characteristic domain="daylightSavingTimeIndicator"
value="true"/>
                        </PropertyValue>
                    <PropertyValue name="AmountSpecification">
                        <Characteristic domain="Amount" value="200" code="USD"/>
                        <Characteristic domain="LowerAmount" value="100"
code="EUR"/>
                        <Characteristic domain="UpperAmount" value="300"
code="EUR"/>
                    </PropertyValue>
                    <PropertyValue name="QuantitySpecification">
                        <Characteristic domain="Quantity" value="200"
code="BOX"/>
                        <Characteristic domain="LowerQuantity" value="100"
code="EA"/>
                        <Characteristic domain="UpperQuantity" value="300"
code="EA"/>
                    </PropertyValue>
                </ValueGroup>
            </PropertyValuation>
        </Batch>
    </SubcontractingComponent>
    <SubcontractingComponent quantity="2"
requirementDate="2013-08-02T14:37:31-07:00">
        <ComponentID>2</ComponentID>
        <UnitOfMeasure>BOX</UnitOfMeasure>
        <Description xml:lang="en">Tailpiece</Description>
        <Product>
            <SupplierPartID>SupplierPartID2</SupplierPartID>
            <SupplierPartAuxiliaryID>SupplierPartAuxiliaryID2</
SupplierPartAuxiliaryID>
            <BuyerPartID>BuyerPartID2</BuyerPartID>
            <IdReference domain="StandardID" identifier="1000"/>
            <IdReference domain="InternalID" identifier="2000"/>
        </Product>
        <ProductRevisionID>12345</ProductRevisionID>
        <Batch originCountryCode="US"
expirationDate="2014-06-10T14:37:31-07:00"
productionDate="2013-06-10T14:37:31-07:00">
            <BuyerBatchID>BuyerBatchID2</BuyerBatchID>
            <SupplierBatchID>SupplierBatchID2</SupplierBatchID>
        </Batch>
    </SubcontractingComponent>
    <PropertyValuation>
        <PropertyReference>
            <IdReference domain="ID" identifier="CHEMICAL"/>
        </PropertyReference>
    </ValueGroup>
        <IdReference domain="ID" identifier="2"/>
    </ValueGroup>
</PropertyValuation>
<PropertyValuation>
    <PropertyReference>
        <IdReference domain="ID" identifier="DILUTION"/>
    </PropertyReference>
</PropertyValuation>

```

```

        </PropertyReference>
        <ValueGroup>
            <IdReference domain="ID" identifier="dilu"/>
            <ParentID>2</ParentID>
            <PropertyValue name="NameSpecification">
                <Characteristic domain="Name" value="1+0 - 1+3"/>
            </PropertyValue>
        </ValueGroup>
    </PropertyValuation>
    <PropertyValuation>
        <PropertyReference>
            <IdReference domain="ID" identifier="CAPACITY"/>
        </PropertyReference>
        <ValueGroup>
            <IdReference domain="ID" identifier="cap"/>
            <ParentID>2</ParentID>
            <PropertyValue name="NameSpecification">
                <Characteristic domain="Name" value="1-4 lt"/>
            </PropertyValue>
        </ValueGroup>
    </PropertyValuation>
    <PropertyValuation>
        <PropertyReference>
            <IdReference domain="ID" identifier="CHEM_FORM"/>
        </PropertyReference>
        <ValueGroup>
            <IdReference domain="ID" identifier="form"/>
            <ParentID>2</ParentID>
            <PropertyValue name="NameSpecification">
                <Characteristic domain="Name" value="flussig"/>
                <Characteristic domain="languageCode" value="de"/>
            </PropertyValue>
            <PropertyValue name="NameSpecification">
                <Characteristic domain="Name" value="liquid"/>
                <Characteristic domain="languageCode" value="en"/>
            </PropertyValue>
        </ValueGroup>
    </PropertyValuation>
</Batch>
</SubcontractingComponent>
<SubcontractingComponent quantity="3"
requirementDate="2013-08-03T14:37:31-07:00">
    <ComponentID>3</ComponentID>
    <UnitOfMeasure>BOX</UnitOfMeasure>
    <Description xml:lang="en">Bridge</Description>
    <Product>
        <SupplierPartID>SupplierPartID3</SupplierPartID>
        <SupplierPartAuxiliaryID>SupplierPartAuxiliaryID3</
SupplierPartAuxiliaryID>
        <BuyerPartID>BuyerPartID3</BuyerPartID>
        <IdReference domain="StandardID" identifier="1000"/>
        <IdReference domain="InternalID" identifier="2000"/>
    </Product>
    <ProductRevisionID>12345</ProductRevisionID>
    <Batch originCountryCode="US"
expirationDate="2014-06-10T14:37:31-07:00"
productionDate="2013-06-10T14:37:31-07:00">
        <BuyerBatchID>BuyerBatchID3</BuyerBatchID>
        <SupplierBatchID>SupplierBatchID3</SupplierBatchID>
    </Batch>
</SubcontractingComponent>
</ScheduleLine>
<Batch originCountryCode="US" expirationDate="2014-06-10T14:37:31-07:00"
productionDate="2013-06-10T14:37:31-07:00">
    <BuyerBatchID>BuyerBatchID1-Item1</BuyerBatchID>
    <SupplierBatchID>SupplierBatchID1-Item1</SupplierBatchID>
</Batch>
</ItemOut>

```

## Incoterms on orders

Incoterms are a series of predefined commercial terms published by the International Chamber of Commerce (ICC). The Incoterms three-letter trade terms are related to common contractual sales practices. They are intended primarily to clearly communicate the tasks, costs, and risks associated with the transportation and delivery of goods. The Incoterms 2010 edition supports the following 11 rules:

- EXW (Ex Works)
- FCA (Free Carrier)
- CPT (Carriage Paid To)
- CIP (Carriage And Insurance Paid To)
- DAT (Delivered At Terminal)
- DAP (Delivered At Place)
- DDP (Delivered Duty Paid)
- FAS (Free Alongside Ship)
- FOB (Free On Board)
- CFR (Cost and Freight)
- CIF (Cost, Insurance and Freight)

For detailed descriptions of the Incoterms rules, see the [International Chamber of Commerce website](#).

To support Incoterms in orders at the header and line-item level of orders, the following cXML extrinsics have been added to `OrderRequestHeader` and `ItemDetail`:

Extrinsic Name	This Field Stores...
<code>incoTerm</code>	Three-letter Incoterms code
<code>incoTermDesc</code>	Incoterms description
<code>incoTermLocation</code>	Incoterms location

The following cXML excerpt shows extrinsic elements for Incoterms in `OrderRequestHeader`:

```
<OrderRequestHeader orderDate="2016-02-16T01:16:58-08:00"
  orderID="po-it453245" orderType="regular" type="new">
  ...
  <Extrinsic name = "incoTerm">FOB</Extrinsic>
  <Extrinsic name = "incoTermDesc">Free on Board</Extrinsic>
  <Extrinsic name = "incoTermLocation">Lima</Extrinsic>
</OrderRequestHeader>
```

The following cXML excerpt shows extrinsic elements for Incoterms in `ItemDetail`:

```
<ItemDetail>
  <UnitPrice>
    <Money currency="USD">20.00</Money>
  </UnitPrice>
  <Description xml:lang="en">Computer Video Cables</Description>
  <UnitOfMeasure>EA</UnitOfMeasure>
  ...
  <Extrinsic name = "incoTerm">EXW</Extrinsic>
  <Extrinsic name = "incoTermDesc">Ex Works</Extrinsic>
  <Extrinsic name = "incoTermLocation">Santiago</Extrinsic>
</ItemDetail>
```



## AribaNetwork.PaymentTermsExplanation extrinsic

A cXML extrinsic field supports the payment terms explanation, which is a freeform description of the payment terms received from the buyer's system. If the supplier is not allowed to change the payment terms, the payment terms explanation is flipped from the order to the invoice for information purposes.

The following cXML extrinsic is supported in `OrderRequestHeader` and `InvoiceDetailRequestHeader`:

Extrinsic Name	This Field Stores...
<code>AribaNetwork.PaymentTermsExplanation</code>	Free-form text describing the payment terms for information purposes.

The following cXML excerpt shows the payment terms explanation in the header of an order:

```
<Request deploymentMode="production">
  <OrderRequest>
    <OrderRequestHeader orderDate="2016-01-29T01:16:58-08:00"
      orderID="po-ptc20150129" orderType="regular" type="new">
      ...
      <PaymentTerm payInNumberOfDays = "14">
        <Discount>
          <DiscountPercent percent = "1.000"></DiscountPercent>
        </Discount>
      </PaymentTerm>
      <PaymentTerm payInNumberOfDays = "30">
      </PaymentTerm>
      <Extrinsic name="AribaNetwork.PaymentTermsExplanation">
        After receipt of invoice: 14days 1% disc. 30days net
      </Extrinsic>
    </OrderRequestHeader>
  </OrderRequest>
</Request>
```

### Note

The `AribaNetwork.PaymentTermsExplanation` extrinsic can appear in the `OrderRequestHeader` or `InvoiceDetailRequestHeader` without any accompanying `PaymentTerm` elements.

## Schedule agreement releases

The cXML 1.2.026 DTD introduced changes that support schedule agreement releases.

For information about cXML, see the DTD and *cXML User's Guide* available at <http://www.cxml.org>. Also, see the *cXML Solutions Guide* on Ariba Network.

## ReleaseInfo

`ReleaseInfo` has been added to `ItemOut` as an optional element. `ReleaseInfo` stores the details about a release of items or materials.

`ReleaseInfo` may contain the following elements:

Element	Description
<code>ShipNoticeReleaseInfo</code>	References the previous ship notice created from a delivery schedule. This reference is against the previous shipment made for the schedule line in the schedule agreement release.
<code>UnitofMeasure</code>	Unit of measure for the quantity specified for the schedule line item.
<code>Extrinsic</code>	Any additional information for the schedule line item.

`ReleaseInfo` may contains the following attributes:

Attribute	Description
<code>releaseType</code>	A mandatory field. A string value to identify the type of delivery schedule against the schedule agreement release.  Possible values: <ul style="list-style-type: none"><li>• <b>JIT</b> (Just-In-Time)</li><li>• <b>forecast</b></li></ul>
<code>cumulativeReceivedQuantity</code>	A mandatory field. A number value to identify the cumulative quantity of all goods received against the scheduling agreement release over a period up to a certain date.
<code>productionGoAheadEndDate</code>	An optional field. Date denoting the end of the production go-ahead period (go-ahead for production).
<code>materialGoAheadEndDate</code>	Date denoting the end of the material go-ahead period (go-ahead for purchase of input materials).

### Example

```
<ReleaseInfo releaseType="forecast" cumulativeReceivedQuantity="0"
  productionGoAheadEndDate="2014-06-13T14:37:31-07:00"
  materialGoAheadEndDate="2014-11-14T14:37:31-07:00"
  <UnitOfMeasure>EA</UnitOfMeasure>
</ReleaseInfo>
```

## ScheduleLineReleaseInfo

An element, `ScheduleLineReleaseInfo`, has been added to the `ScheduleLine` element.

`ScheduleLineReleaseInfo` stores details about a specific release of items or materials for a schedule line.

`ScheduleLineReleaseInfo` contains the following attributes:

Attribute	Description
<code>commitmentCode</code>	A string value to identify the type of the delivery. The value can be any of the following:  <code>firmg</code> : Go-ahead for production. Vendor can ship against the schedule line. Customer is responsible for cost of production as well as cost of material procurement.  <code>tradeoff</code> : Go-ahead for material procurement. Vendor can ship against the schedule line if rule is enabled. Buyer is responsible for cost of material procurement.  <code>forecast</code> : Informational. Customer can change the schedule line without incurring any liabilities with the vendor.
<code>cumulativeScheduledQuantity</code>	Total quantity to be shipped for a particular line item up through the schedule line.

### Example

```
<ScheduleLine quantity="100" requestedDeliveryDate="2014-05-10T14:37:31-07:00">  
  <UnitOfMeasure>EA</UnitOfMeasure>  
  <ScheduleLineReleaseInfo commitmentCode="firm"  
cumulativeScheduledQuantity="100">  
    <UnitOfMeasure>EA</UnitOfMeasure>  
  </ScheduleLineReleaseInfo>  
</ScheduleLine>
```

## agreementType

An attribute, `agreementType`, has been added to the `MasterAgreementIDInfo` element and

`MasterAgreementReference` element to indicate whether the referenced agreement is a scheduling agreement release.

### Example

```
<MasterAgreementIDInfo agreementID="SA301" agreementType="scheduling_agreement"/>
```

## ShipNoticePortion

ShipNoticePortion may include two additional, optional elements, MasterAgreementReference, and MasterAgreementID to identify the master agreement from which the ship notice is derived.

The following table describes these elements.

Element	Description
MasterAgreementReference	Optional. References the master agreement from which the release is derived.
MasterAgreementIDInfo	Optional. Indicates the ID of the master agreement from which the release is derived.

## Planned and unplanned service line items

Ariba Network supports both planned and unplanned service line items in purchase orders.

### Planned service line items

Planned service line items describe specific services or material goods using <ItemDetail>.

- If planned service line items are individual line items, they require a service sheet.
- If planned services line items are parent item groups with grouped items under them, the parent item group requires a service sheet.

### Example of cXML from the SAP ERP

```
<ItemOut itemType = "composite" lineNumber = "10"
  quantity = "1.0" requestedDeliveryDate = "2016-01-12T12:00:00+01:00"
  requiresServiceEntry = "yes">
  <ItemID>
    <SupplierPartID></SupplierPartID>
  </ItemID>
  <BlanketItemDetail>
    <Description xml:lang = "en">Cleaning Services</Description>
    <MaxAmount>
      <Money currency = "EUR">474.0</Money>
    </MaxAmount>
    <MaxQuantity>1.0</MaxQuantity>
    <MinQuantity>1.0</MinQuantity>
    <UnitPrice>
      <Money currency = "EUR">474.0</Money>
    </UnitPrice>
    <UnitOfMeasure>C62</UnitOfMeasure>
    <PriceBasisQuantity conversionFactor = "1" quantity = "1.0">
      <UnitOfMeasure>C62</UnitOfMeasure>
    </PriceBasisQuantity>
  </BlanketItemDetail>
</ItemOut>
```

```

        </PriceBasisQuantity>
        <Classification domain = "not available">Material
          group 1</Classification>
      </BlanketItemDetail>
      <SpendDetail>
        <Extrinsic name = "ConsiderServiceAsMaterialType"></Extrinsic>
      </SpendDetail>
      <Tolerances>
        <PriceTolerance>
          <Percentage percent = "20.00"></Percentage>
        </PriceTolerance>
      </Tolerances>
      <ScheduleLine lineNumber = "1" quantity = "1.0"
        requestedDeliveryDate = "2016-01-12T12:00:00+01:00">
        <UnitOfMeasure>C62</UnitOfMeasure>
      </ScheduleLine>
    </ItemOut>
    <ItemOut lineNumber = "1000100010" parentLineNumber = "10"
      quantity = "34.0">
      <ItemID>
        <SupplierPartID>SUPPLIER_NO</SupplierPartID>
        <BuyerPartID>1000020</BuyerPartID>
      </ItemID>
      <ItemDetail>
        <UnitPrice>
          <Money currency = "EUR">11.0</Money>
        </UnitPrice>
        <Description xml:lang = "en">Cleaning Windows</Description>
        <UnitOfMeasure>HUR</UnitOfMeasure>
        <Classification domain = "not available">Material group 1
        </Classification>
      </ItemDetail>
      <SpendDetail>
        <Extrinsic name = "ConsiderServiceAsMaterialType"></Extrinsic>
      </SpendDetail>
    </ItemOut>

```

## Example of cXML from the SAP Business Suite Add-On for Ariba Network Integration

```

<ItemOut quantity="1.0" lineNumber="10"
  requestedDeliveryDate="2016-07-13T12:00:00+02:00" itemType="composite"
  requiresServiceEntry="yes">
  <ItemID>
    <SupplierPartID/>
  </ItemID>
  <BlanketItemDetail>
    <Description xml:lang="en">Cleaning Services</Description>
    <MaxAmount>
      <Money currency="EUR">984.0</Money>
    </MaxAmount>
    <MaxQuantity>1.0</MaxQuantity>
    <MinQuantity>1.0</MinQuantity>
    <UnitPrice>
      <Money currency="EUR">984.0</Money>
    </UnitPrice>
    <UnitOfMeasure>C62</UnitOfMeasure>
    <PriceBasisQuantity quantity="1.0" conversionFactor="1">
      <UnitOfMeasure>C62</UnitOfMeasure>
    </PriceBasisQuantity>
    <Classification domain="not available">Material
      group 2</Classification>
  </BlanketItemDetail>
  <SpendDetail>

```

```

    <Extrinsic name="GenericServiceCategory"/>
  </SpendDetail>
  <ScheduleLine quantity="1.0"
    requestedDeliveryDate="2016-07-13T12:00:00+02:00"
    lineNumber="1">
    <UnitOfMeasure>C62</UnitOfMeasure>
  </ScheduleLine>
</ItemOut>
<ItemOut quantity="123.0" lineNumber="1000100010" parentLineNumber="10">
  <ItemID>
    <SupplierPartID/>
    <BuyerPartID>1000020</BuyerPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="EUR">8.0</Money>
    </UnitPrice>
    <Description xml:lang="en">cleaning windows</Description>
    <UnitOfMeasure>HUR</UnitOfMeasure>
    <Classification domain="not available">Material
      group 2</Classification>
    </ItemDetail>
  <SpendDetail>
    <Extrinsic name="GenericServiceCategory">Standard Service</Extrinsic>
  </SpendDetail>
</ItemOut>
<ItemOut
  quantity="1.0"
  lineNumber="20"
  requestedDeliveryDate="2016-07-13T12:00:00+02:00"
  itemType="composite"
  requiresServiceEntry="yes">
  <ItemID>
    <SupplierPartID/>
  </ItemID>
  <BlanketItemDetail>
    <Description xml:lang="en">Maintenance</Description>
    <MaxAmount>
      <Money currency="EUR">800.0</Money>
    </MaxAmount>
    <MaxQuantity>1.0</MaxQuantity>
    <MinQuantity>1.0</MinQuantity>
    <UnitPrice>
      <Money currency="EUR">800.0</Money>
    </UnitPrice>
    <UnitOfMeasure>C62</UnitOfMeasure>
    <PriceBasisQuantity quantity="1.0" conversionFactor="1">
      <UnitOfMeasure>C62</UnitOfMeasure>
    </PriceBasisQuantity>
    <Classification domain="not available">Material
      group 2</Classification>
    </BlanketItemDetail>
  <SpendDetail>
    <Extrinsic name="GenericServiceCategory"/>
  </SpendDetail>
  <ScheduleLine
    quantity="1.0"
    requestedDeliveryDate="2016-07-13T12:00:00+02:00"
    lineNumber="1">
    <UnitOfMeasure>C62</UnitOfMeasure>
  </ScheduleLine>
</ItemOut>
<ItemOut
  quantity="1.0"
  lineNumber="1000200010"
  parentLineNumber="20">
  <ItemID>
    <SupplierPartID/>

```

```

    <BuyerPartID>1000030</BuyerPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="EUR">800.0</Money>
    </UnitPrice>
    <Description xml:lang="en">yearly maintenance
      central heating</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="not available">Material
      group 2</Classification>
  </ItemDetail>
  <SpendDetail>
    <Extrinsic name="GenericServiceCategory">Maintenance</Extrinsic>
  </SpendDetail>
</ItemOut>

```

## Unplanned service line items

Unplanned service line items describe general services using `<BlanketItemDetail>` and a required `<MaxAmount>`.

Unplanned service lines have the following characteristics:

- If they are individual lines, they require a service sheet.
- If they are parent item groups, they do not contain any grouped items under them.

### Example

```

<ItemOut lineNumber="1" quantity="1" requestedDeliveryDate="2013-09-1"
requiresServiceEntry="yes">
  <ItemID>
    <SupplierPartID>LW4001</SupplierPartID>
    <SupplierPartAuxiliaryID></SupplierPartAuxiliaryID>
  </ItemID>
  <BlanketItemDetail>
    <Description xml:lang="en">Site preparation of parcel no. 110
      (3 acres)</Description>
    <MaxAmount>
      <Money currency="USD">70000</Money>
    </MaxAmount>
    <UnitPrice><Money currency="USD">50000</Money></UnitPrice>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="ascc">606501</Classification>
    <Extrinsic name="Construction"></Extrinsic>
    <Extrinsic name="ServicePeriod">
      <Period endDate="2013-09-01T23:59:59-00:00"
        startDate="2012-07-12T00:00:00-00:00">
      </Period>
    </Extrinsic>
  </BlanketItemDetail>
  <Distribution>
    <Accounting name="Distribution Charge">
      <AccountingSegment id="100">
        <Name xml:lang="en">Construction</Name>
        <Description xml:lang="en">Western Division</Description>
      </AccountingSegment>
    </AccountingSegment>
  </Distribution>
</ItemOut>

```

```

    </Accounting>
    <Charge><Money currency="USD">50000</Money></Charge>
  </Distribution>
</ItemOut>

```

## ExpectedUnplanned and MaximumUnplanned Extrinsics

You can pass **Expected Value** and **Overall Limit** values for unplanned line items by using extrinsics.

**Expected Value** is displayed to both buyers and suppliers on purchase orders at the line-item level. **Overall Limit** is not exposed to supplier users. Suppliers cannot submit service sheets that include unplanned line items with a value that exceeds the **Overall Limit**.

### Note

**Overall Limit** must be greater than or equal to **Expected Value**.

The following extrinsic elements support **Expected Value** and **Overall Limit**:

Element	Contained in	Description
<Extrinsic name="ExpectedUnplanned">	<BlanketItemDetail>	Expected value for unplanned line items. Displayed to buyer and supplier on purchase order.
<Extrinsic name="MaximumUnplanned">	<BlanketItemDetail>	Overall limit (maximum value) for unplanned line items. Not disclosed to suppliers.

cXML service orders and service sheets downloaded from the Ariba Network user interface by suppliers do not contain MaximumUnplanned extrinsic elements.

## Example cXML with Expected Value and Overall Limit

```

<ItemOut lineNumber="1" quantity="1"
  requestedDeliveryDate="2015-12-17" requiresServiceEntry="yes">
  <ItemID>
    <SupplierPartID>LW4001</SupplierPartID>
    <SupplierPartAuxiliaryID/>
  </ItemID>
  <BlanketItemDetail>
    <Description xml:lang="en">Site preparation of parcel no. 100
      (3 acres)</Description>
    <UnitPrice>
      <Money currency="USD">50000</Money>
    </UnitPrice>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="ascc">606501</Classification>
    <Extrinsic name="ServicePeriod">
      <Period endDate="2015-12-17T17:00:00+08:00"
        startDate="2015-09-23T08:00:00+08:00"/>
    </Extrinsic>
    <Extrinsic name="ExpectedUnplanned">
      <Money currency="USD">50000</Money>
    </Extrinsic>
  </BlanketItemDetail>
</ItemOut>

```



```
<Extrinsic name="MaximumUnplanned">
  <Money currency="USD">70000</Money>
</Extrinsic>
</BlanketItemDetail>
</ItemOut>
```

## Planned vs. unplanned service lines

When you create cXML purchase order applications, you may need to distinguish between planned and unplanned service lines.

The following table summarizes the characteristics of planned vs. unplanned service lines:

Service Line	Characteristics	Examples
Planned	<p>The service line is an item group with grouped items under it. If the unit price is visible, the subtotals of the grouped item add up to the unit price.</p> <p>If the buyer allows suppliers to add ad hoc items to an order and the service line specifies a (hidden) maximum amount that is larger than the unit price, the supplier can add unplanned/ad hoc items to a planned service line up to tolerance limits.</p>	<p>Planned line:</p> <ul style="list-style-type: none"> <li>1 Service Line A \$300 (max amount \$300) <ul style="list-style-type: none"> <li>1.1 Service Item A1 \$200</li> <li>1.2 Goods Item A2 \$100</li> </ul> Planned line to which supplier can add unplanned/ad hoc items: </li> <li>2 Service Line B \$300 (max amount \$500) <ul style="list-style-type: none"> <li>2.1 Service Item B1 \$200</li> <li>2.2 Goods Item B2 \$100</li> </ul> </li> </ul>
Unplanned	<p>The service line is an item group with no items grouped under it. The unit price might be visible or hidden.</p> <p>The supplier must add items to the unplanned service line. The subtotal of those items can add up to the unit price, if it is visible, and must not exceed the (hidden) maximum amount.</p>	<p>Unplanned line with unit price:</p> <ul style="list-style-type: none"> <li>1 Service Line A \$300 (max amount \$300)</li> </ul> <p>Unplanned line with hidden amount:</p> <ul style="list-style-type: none"> <li>2 Service Line B undisclosed (max amount \$500)</li> </ul>

## Extrinsic for clickable links

Ariba Network can render URLs as clickable links in online purchase orders and invoices.

Clickable URL links are produced by `Extrinsic` elements of the form:

```
<Extrinsic name="anyname">
  <URL name="click me">http://www.bigcompany.com/info</URL>
</Extrinsic>
```

This example produces the following online text:

```
anyname:click me
```

The words “click me” are underlined and are clickable, and the link destination is `http://www.bigcompany.com/info</URL>`.

---

If the URL element has no `name` attribute, Ariba Network displays the URL and makes it clickable.

## Extrinsics for service periods

You can specify service periods for service lines in a purchase order by using the `ServicePeriod` extrinsic in the `GenericServiceCategory` extrinsic in either `ItemDetail` or `BlanketItemDetail`. Alternatively, you can specify the `ServicePeriod` within the `SpendDetail` element in `ItemOut`. Both are valid cXML. Whichever method you use, make sure the Ariba Network adapter for your back-end system maps the data appropriately.

### **i** Note

SAP Business Suite Add-On for Ariba Network Integration uses `ScheduleLine@requestedDeliveryDate` to specify the requested delivery date for line items.

### Example of `ServicePeriod` within `GenericServiceCategory`

```
<Extrinsic name="GenericServiceCategory">
  <Extrinsic name="ServicePeriod">
    <Period endDate="2013-09-01T23:59:59-00:00"
      startDate="2012-07-12T00:00:00-00:00"></Period>
  </Extrinsic>
</Extrinsic>
```

### Example of `ServicePeriod` within `SpendDetail`

```
<SpendDetail>
  <Extrinsic name="ServicePeriod">
    <Period endDate="2013-09-01T23:59:59-00:00"
      startDate="2012-07-12T00:00:00-00:00"></Period>
  </Extrinsic>
</SpendDetail>
```

## Related Information

[ScheduleLine element \[page 153\]](#)

## 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.

### Example

```
<OrderRequestHeader orderDate="2015-10-31T13:04:04-08:00" 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/New."

## Changes introduced by Ariba Buyer 7.0.6

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"
```

## External line numbers

To support external line numbers of items in purchase orders, the following cXML extrinsic has been added to `BlanketItemDetail` and `ItemDetail`:

Extrinsic Name	This field stores...
<code>extLineNumber</code>	A number. It can have leading zeroes (for example, "001"). When the Ariba Network displays the external line number for an item, it concatenates all external line numbers for the item and all its parent items.

The following cXML excerpt shows the new extrinsic element for purchase order external line items:

```
<BlanketItemDetail>
  <Description
    xml:lang="en">ADB2226 - Build a house</Description>
  <UnitPrice>
    <Money currency="USD">50000.00</Money>
  </UnitPrice>
  <UnitOfMeasure>EA</UnitOfMeasure>
  <Classification domain="UNSPSC">721315</Classification>
  ...
  <Extrinsic name="extLineNumber"/>001</Extrinsic>
</BlanketItemDetail>
```

## Service order with multi-level hierarchy

A service order with a multi-level hierarchy has the following characteristics:

- Top-level outline items have `MaxAmount`, `UnitPrice`, `UnitOfMeasure`, and `PriceBasisQuantity` elements. Outline items that are not at the top level do NOT have these elements.
- The `ItemOut` element can contain a `requiresServiceEntry="yes"` attribute, which specifies that the supplier must create a service sheet for the line item.
- The `parentLineNumber` attribute is a mandatory field for all items with parent items. It refers to the line number of the parent line item.
- The `ItemType` attribute can contain two values: "composite" to identify an outline item or "item" to identify an independent line item. `ItemOut` elements with either `ItemType` can require a service entry sheet.
- Items have external line numbers specified by the `extLineNumber` extrinsic element.

### Example

```
<OrderRequest>
  <OrderRequestHeader orderID = "4500018482" orderType = "regular" orderDate =
    "2015-03-31T02:30:01+05:30" type = "new">
    <Total>
      <Money currency = "USD">239700.00</Money>
    </Total>
    <ShipTo>
      <Address isoCountryCode = "US" addressID = "3000">
        <Name xml:lang = "EN">New York</Name>
```

```

        <PostalAddress>
            <Street>691 Broadway</Street>
            <City>NEW YORK</City>
            <Municipality>NEW YORK</Municipality>
            <State>NY</State>
            <PostalCode>10001</PostalCode>
            <Country isoCountryCode = "US"></Country>
        </PostalAddress>
        <Phone>
            <TelephoneNumber>
                <CountryCode isoCountryCode = "US"></CountryCode>
                <AreaOrCityCode></AreaOrCityCode>
                <Number>001-9287-34571</Number>
            </TelephoneNumber>
        </Phone>
        <Fax>
            <TelephoneNumber>
                <CountryCode isoCountryCode = "US"></CountryCode>
                <AreaOrCityCode></AreaOrCityCode>
                <Number>001-9287-34573</Number>
            </TelephoneNumber>
        </Fax>
    </Address>
</ShipTo>
<BillTo>
    <Address isoCountryCode = "US" addressID = "3000">
        <Name xml:lang = "EN">IDES US INC</Name>
        <PostalAddress>
            <Street>1230 Lincoln Avenue</Street>
            <City>NEW YORK</City>
            <Municipality>NEW YORK</Municipality>
            <State>NY</State>
            <PostalCode>10019</PostalCode>
            <Country isoCountryCode = "US"></Country>
        </PostalAddress>
        <Phone>
            <TelephoneNumber>
                <CountryCode isoCountryCode = "US"></CountryCode>
                <AreaOrCityCode></AreaOrCityCode>
                <Number>212-555-0983</Number>
            </TelephoneNumber>
        </Phone>
        <Fax>
            <TelephoneNumber>
                <CountryCode isoCountryCode = "US"></CountryCode>
                <AreaOrCityCode></AreaOrCityCode>
                <Number>212-555-5693</Number>
            </TelephoneNumber>
        </Fax>
    </Address>
</BillTo>
<PaymentTerm payInNumberOfDays = "14">
    <Discount>
        <DiscountPercent percent = "3.000"></DiscountPercent>
    </Discount>
</PaymentTerm>
<PaymentTerm payInNumberOfDays = "30">
    <Discount>
        <DiscountPercent percent = "2.000"></DiscountPercent>
    </Discount>
</PaymentTerm>
<PaymentTerm payInNumberOfDays = "45"></PaymentTerm>
<Contact role = "supplierCorporate" addressID = "0000001000">
    <Name xml:lang = "EN">C.E.B. BERLIN</Name>
    <PostalAddress>
        <Street>Kolping Str. 15</Street>
        <City>Berlin</City>
        <Municipality></Municipality>
    </PostalAddress>

```

```

        <State>11</State>
        <PostalCode>12001</PostalCode>
        <Country isoCountryCode = "DE"></Country>
    </PostalAddress>
    <Email></Email>
    <Phone>
        <TelephoneNumber>
            <CountryCode isoCountryCode = "DE"></CountryCode>
            <AreaOrCityCode></AreaOrCityCode>
            <Number>06894/55501-0</Number>
        </TelephoneNumber>
    </Phone>
    <Fax>
        <TelephoneNumber>
            <CountryCode isoCountryCode = "DE"></CountryCode>
            <AreaOrCityCode></AreaOrCityCode>
            <Number>06894/55501-100</Number>
        </TelephoneNumber>
    </Fax>
</Contact>
<Extrinsic name = "CompanyCode">3000</Extrinsic>
<Extrinsic name = "Ariba.invoicingAllowed">Yes</Extrinsic>
<Extrinsic name = "Ariba.availableAmount">239700</Extrinsic>
<Extrinsic name = "partyAdditionalID">0000001000</Extrinsic>
</OrderRequestHeader>
<ItemOut itemType = "composite" requestedDeliveryDate =
    "2015-04-27T00:00:00+05:30" lineNumber = "00010" quantity = "1.000"
    requiresServiceEntry = "yes">
    <ItemID>
        <SupplierPartID></SupplierPartID>
        <BuyerPartID></BuyerPartID>
    </ItemID>
    <BlanketItemDetail>
        <Description xml:lang = "en">SAP Consulting</Description>
        <MaxAmount>
            <Money currency = "USD">239700</Money>
        </MaxAmount>
        <UnitPrice>
            <Money currency = "USD">239700</Money>
        </UnitPrice>
        <UnitOfMeasure>SU</UnitOfMeasure>
        <PriceBasisQuantity conversionFactor = "1" quantity = "1">
            <UnitOfMeasure>SU</UnitOfMeasure>
        </PriceBasisQuantity>
        <Classification domain = "NotAvailable">00801</Classification>
        <Extrinsic name = "AccountCategory">K</Extrinsic>
        <Extrinsic name = "ReceivingType"></Extrinsic>
        <Extrinsic name = "extLineNumber">00010</Extrinsic>
    </BlanketItemDetail>
    <ScheduleLine requestedDeliveryDate = "2015-04-27T00:00:00+05:30"
        quantity = "1.000">
        <UnitOfMeasure>SU</UnitOfMeasure>
    </ScheduleLine>
</ItemOut>
<ItemOut parentLineNumber = "00010" itemType = "composite"
    lineNumber = "0000200010" quantity = "1.0" requiresServiceEntry = "yes">
    <ItemID>
        <SupplierPartID></SupplierPartID>
    </ItemID>
    <BlanketItemDetail>
        <Description xml:lang = "EN">Functional Architect</Description>
        <Extrinsic name = "AccountCategory">K</Extrinsic>
        <Extrinsic name = "extLineNumber">FUN_ARCH</Extrinsic>
    </BlanketItemDetail>
    <SpendDetail>
        <Extrinsic name = "service">service</Extrinsic>
    </SpendDetail>
</ItemOut>

```

```

<ItemOut parentLineNumber = "0000200010" itemType = "composite"
  lineNumber = "0000300010" quantity = "1.0" requiresServiceEntry = "yes">
  <ItemID>
    <SupplierPartID></SupplierPartID>
  </ItemID>
  <BlanketItemDetail>
    <Description xml:lang = "EN">Technical Consultants</Description>
    <Extrinsic name = "AccountCategory">K</Extrinsic>
    <Extrinsic name = "extLineNumber">CONSULTS</Extrinsic>
  </BlanketItemDetail>
  <SpendDetail>
    <Extrinsic name = "service">service</Extrinsic>
  </SpendDetail>
</ItemOut>
<ItemOut parentLineNumber = "00010" lineNumber = "1000100010"
  quantity = "45.000">
  <ItemID>
    <SupplierPartID></SupplierPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency = "USD">1500</Money>
    </UnitPrice>
    <Description xml:lang = "EN">Architects</Description>
    <UnitOfMeasure>DAY</UnitOfMeasure>
    <PriceBasisQuantity conversionFactor = "1" quantity = "1">
      <UnitOfMeasure>DAY</UnitOfMeasure>
    </PriceBasisQuantity>
    <Classification domain = "UNSPSC">00801</Classification>
    <Extrinsic name = "AccountCategory">K</Extrinsic>
    <Extrinsic name = "extLineNumber">10</Extrinsic>
  </ItemDetail>
  <SpendDetail>
    <Extrinsic name = "service">service</Extrinsic>
  </SpendDetail>
  <Distribution>
    <Accounting name = "DistributionCharge">
      <AccountingSegment id = "0000400000">
        <Name xml:lang = "en">GeneralLedger</Name>
        <Description xml:lang = "en"></Description>
      </AccountingSegment>
      <AccountingSegment id = "0000002100">
        <Name xml:lang = "en">CostCenter</Name>
        <Description xml:lang = "en"></Description>
      </AccountingSegment>
      <AccountingSegment id = "100.00">
        <Name xml:lang = "en">Percentage</Name>
        <Description xml:lang = "en"></Description>
      </AccountingSegment>
    </Accounting>
    <Charge>
      <Money currency = "USD">67500.00</Money>
    </Charge>
  </Distribution>
</ItemOut>
<ItemOut parentLineNumber = "0000200010" lineNumber =
  "1000200010" quantity = "90.000">
  <ItemID>
    <SupplierPartID></SupplierPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency = "USD">900</Money>
    </UnitPrice>
    <Description xml:lang = "EN">Functional Consultants</Description>
    <UnitOfMeasure>DAY</UnitOfMeasure>
    <PriceBasisQuantity conversionFactor = "1" quantity = "1">
      <UnitOfMeasure>DAY</UnitOfMeasure>
    </PriceBasisQuantity>
  </ItemDetail>

```

```

        </PriceBasisQuantity>
        <Classification domain = "UNSPSC">00801</Classification>
        <Extrinsic name = "AccountCategory">K</Extrinsic>
        <Extrinsic name = "extLineNumber">10</Extrinsic>
    </ItemDetail>
    <SpendDetail>
        <Extrinsic name = "service">service</Extrinsic>
    </SpendDetail>
    <Distribution>
        <Accounting name = "DistributionCharge">
            <AccountingSegment id = "0000400000">
                <Name xml:lang = "en">GeneralLedger</Name>
                <Description xml:lang = "en"></Description>
            </AccountingSegment>
            <AccountingSegment id = "0000001200">
                <Name xml:lang = "en">CostCenter</Name>
                <Description xml:lang = "en"></Description>
            </AccountingSegment>
            <AccountingSegment id = "100.00">
                <Name xml:lang = "en">Percentage</Name>
                <Description xml:lang = "en"></Description>
            </AccountingSegment>
        </Accounting>
        <Charge>
            <Money currency = "USD">81000.00</Money>
        </Charge>
    </Distribution>
</ItemOut>
<ItemOut parentLineNumber = "0000200010" lineNumber =
    "1000200020" quantity = "110.000">
    <ItemID>
        <SupplierPartID></SupplierPartID>
    </ItemID>
    <ItemDetail>
        <UnitPrice>
            <Money currency = "USD">750</Money>
        </UnitPrice>
        <Description xml:lang = "EN">Technical Architect</Description>
        <UnitOfMeasure>DAY</UnitOfMeasure>
        <PriceBasisQuantity conversionFactor = "1" quantity = "1">
            <UnitOfMeasure>DAY</UnitOfMeasure>
        </PriceBasisQuantity>
        <Classification domain = "UNSPSC">00801</Classification>
        <Extrinsic name = "AccountCategory">K</Extrinsic>
        <Extrinsic name = "extLineNumber">20</Extrinsic>
    </ItemDetail>
    <SpendDetail>
        <Extrinsic name = "service">service</Extrinsic>
    </SpendDetail>
    <Distribution>
        <Accounting name = "DistributionCharge">
            <AccountingSegment id = "0000400000">
                <Name xml:lang = "en">GeneralLedger</Name>
                <Description xml:lang = "en"></Description>
            </AccountingSegment>
            <AccountingSegment id = "0000002100">
                <Name xml:lang = "en">CostCenter</Name>
                <Description xml:lang = "en"></Description>
            </AccountingSegment>
            <AccountingSegment id = "100.00">
                <Name xml:lang = "en">Percentage</Name>
                <Description xml:lang = "en"></Description>
            </AccountingSegment>
        </Accounting>
        <Charge>
            <Money currency = "USD">82500.00</Money>
        </Charge>
    </Distribution>

```



```

</ItemOut>
<ItemOut parentLineNumber = "0000300010" lineNumber =
  "1000300010" quantity = "7.000">
  <ItemID>
    <SupplierPartID></SupplierPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency = "USD">600</Money>
    </UnitPrice>
    <Description xml:lang = "EN">Sr.Consultants</Description>
    <UnitOfMeasure>DAY</UnitOfMeasure>
    <PriceBasisQuantity conversionFactor = "1" quantity = "1">
      <UnitOfMeasure>DAY</UnitOfMeasure>
    </PriceBasisQuantity>
    <Classification domain = "UNSPSC">00801</Classification>
    <Extrinsic name = "AccountCategory">K</Extrinsic>
    <Extrinsic name = "extLineNumber">10</Extrinsic>
  </ItemDetail>
  <SpendDetail>
    <Extrinsic name = "service">service</Extrinsic>
  </SpendDetail>
  <Distribution>
    <Accounting name = "DistributionCharge">
      <AccountingSegment id = "0000400000">
        <Name xml:lang = "en">GeneralLedger</Name>
        <Description xml:lang = "en"></Description>
      </AccountingSegment>
      <AccountingSegment id = "0000002100">
        <Name xml:lang = "en">CostCenter</Name>
        <Description xml:lang = "en"></Description>
      </AccountingSegment>
      <AccountingSegment id = "100.00">
        <Name xml:lang = "en">Percentage</Name>
        <Description xml:lang = "en"></Description>
      </AccountingSegment>
    </Accounting>
    <Charge>
      <Money currency = "USD">4200.00</Money>
    </Charge>
  </Distribution>
</ItemOut>
<ItemOut parentLineNumber = "0000300010" lineNumber =
  "1000300020" quantity = "10.000">
  <ItemID>
    <SupplierPartID></SupplierPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency = "USD">450</Money>
    </UnitPrice>
    <Description xml:lang = "EN">Jr.Consultants</Description>
    <UnitOfMeasure>DAY</UnitOfMeasure>
    <PriceBasisQuantity conversionFactor = "1" quantity = "1">
      <UnitOfMeasure>DAY</UnitOfMeasure>
    </PriceBasisQuantity>
    <Classification domain = "UNSPSC">00801</Classification>
    <Extrinsic name = "AccountCategory">K</Extrinsic>
    <Extrinsic name = "extLineNumber">20</Extrinsic>
  </ItemDetail>
  <SpendDetail>
    <Extrinsic name = "service">service</Extrinsic>
  </SpendDetail>
  <Distribution>
    <Accounting name = "DistributionCharge">
      <AccountingSegment id = "0000400000">
        <Name xml:lang = "en">GeneralLedger</Name>
        <Description xml:lang = "en"></Description>

```

```

        </AccountingSegment>
        <AccountingSegment id = "0000001200">
            <Name xml:lang = "en">CostCenter</Name>
            <Description xml:lang = "en"></Description>
        </AccountingSegment>
        <AccountingSegment id = "100.00">
            <Name xml:lang = "en">Percentage</Name>
            <Description xml:lang = "en"></Description>
        </AccountingSegment>
    </Accounting>
    <Charge>
        <Money currency = "USD">4500.00</Money>
    </Charge>
</Distribution>
</ItemOut>
</OrderRequest>

```

## 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.

### Related Information

[Order confirmations and ship notices \[page 193\]](#)

## Example response document

A simple response to an `OrderRequest` is an HTTP acknowledgement that the order was received.

### Example

```

<?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:

Routing Method	Supplier Acknowledgment	Time out for Acknowledgment	Ariba Network Re-sends PO?
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.

---

# Order confirmations and ship notices

Suppliers can update purchase order status either manually through their Ariba Network accounts or by submitting cXML documents to Ariba Network. 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`.

## In this section:

- [Visibility of order status \[page 194\]](#)
- [Ariba Network account configuration \[page 195\]](#)
- [Ariba Buyer 8.1 and later transaction flow \[page 196\]](#)
- [Overview of `ConfirmationRequest` documents \[page 196\]](#)
- [OrderStatusRequest documents \[page 205\]](#)
- [ShipNoticeRequest documents \[page 207\]](#)
- [Order confirmation and ship notice attachments \[page 215\]](#)

---

## Visibility of order status

Buying organizations, requisitioners, and suppliers can view order status.

### Visibility of order status for buying organizations

Authorized procurement personnel within buying organizations can log in to Ariba Network and view order status.

### Visibility of order status for 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.

---

## Visibility of order status for 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 account configuration

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 account configuration

Before sending `ConfirmationRequest` or `ShipNoticeRequest` documents, suppliers must first configure their Ariba Network accounts to set their preferred method of routing Order Response Documents. Suppliers can set order status and shipping information either manually or automatically.

To set programmatic order status updates:

1. Log into the Ariba Network as a supplier.
2. Click your supplier name, and select **Network Settings > Electronic Order Routing**.
3. On the **Electronic Order Routing** tab, find the **Other Document Types** section.
4. For **Order Response Documents**, select cXML.
5. Click **Save**.

#### Note

Suppliers determine the URL for posting `ConfirmationRequest` and `ShipNoticeRequest` documents by using the Profile transaction. For more information, see [Profile Transaction \[page 63\]](#).

These suppliers can still manually update order status by temporarily changing the routing of **Order 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

## Ariba Buyer 8.1 and later transaction flow

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.

The process is as follows.

1. The supplier sends a `ConfirmationRequest` 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 `ConfirmationRequest` 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`.

For the credential values for order confirmations and ship notices, see [Required credentials \[page 40\]](#).

## Overview of 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

The ConfirmationHeader element describes the type and operation of the confirmation request.

ConfirmationHeader has the following attributes and elements:

Attribute	Description
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).

Attribute	Description
Total	<p>Total dollar value of all line items in the transaction.</p> <p>The <code>Total</code> element also contains the <code>Modifications</code> 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 <code>Modification</code> elements.</p> <p>The <code>Modification</code> element contains details of the allowances and charges applicable at the header-level. For more information, see <a href="#">Modifications Element [page 136]</a>.</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.

Optional display-only elements allowed in `ConfirmationHeader` include:

- `Total`
- `Shipping`
- `Tax`
- `Contact`
- `Comments`
- `Hazard`

## OrderReference element

The `OrderReference` element refers to either the original purchase order or a previous confirmation request.

`OrderReference` has the following attributes:

Attribute	Description
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 <code>ConfirmationRequest</code> is the initial response to an <code>OrderRequest</code> and also at the line item level if the <code>ConfirmationHeader</code> <code>operation</code>="update".</p> <p>For <code>operation</code>="new", the <code>DocumentReference</code> at the header level must refer to the purchase order. For <code>operation</code>="update" or "delete", the <code>DocumentReference</code> at the header level must refer to the previous <code>ConfirmationRequest</code>.</p>

## ConfirmationItem element

The `ConfirmationItem` element indicates the line item number of the item in the `OrderRequest` document.

`ConfirmationItem` has the following attributes:

Attribute	Description
<code>lineNumber</code>	Reference line item number from the <code>OrderRequest</code> document.
<code>quantity</code>	Either original quantity if going to confirm in full or updated quantity with type set appropriately.
<code>Comments</code>	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, <code>shipped</code> and <code>invalid</code> .

## ConfirmationStatus element

The `ConfirmationStatus` element indicates the action taken by a supplier for a specific line item.

`ConfirmationStatus` has the following attributes:

Attribute	Description
<code>quantity</code>	Sum must match the quantity in the <code>ConfirmationItem</code> .
<code>type</code>	Action taken by supplier for line item number identified. Can be: <code>accept</code> , <code>detail</code> , <code>reject</code> , <code>backordered</code> , <code>requestToPay</code> , or <code>unknown</code> . <ul style="list-style-type: none"><li><code>type="accept"</code> accepts particular line item.</li><li><code>type="detail"</code> accepts portion of line item with the changes detailed in the <code>ConfirmationStatus</code> element. At least one of the <code>UnitPrice</code>, <code>Shipping</code>, <code>Tax</code>, or <code>ItemIn</code> elements, or the <code>deliveryDate</code> attribute must be present.</li><li><code>type="reject"</code> rejects this portion of the line item.</li><li><code>type="backordered"</code> 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><code>type="unknown"</code> resets the status to the default state. The item status displays on Ariba Network as if no confirmations had been sent.</li></ul> <code>type="allDetail"</code> is not supported by Ariba Network.
<code>shipmentDate</code>	Date and time this shipment is expected to leave the supplier. Required if type is <code>accept</code> , <code>allDetail</code> , or <code>detail</code> .
<code>deliveryDate</code>	Date and time this shipment is expected to arrive. Required if type is <code>accept</code> , <code>allDetail</code> , or <code>detail</code> .  Do not include if its value matches <code>requestedDeliveryDate</code> (if any) in the purchase order.  Displayed as "Est. Delivery Date" within Ariba Network.

`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 \[page 150\]](#).

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 following elements:

- [Modifications element \[page 136\]](#): The `Modification` element contains details of the allowances and charges applicable at the line-item level.
- `SupplierBatchID`: This element stores the batch ID provided by the supplier at the line item quantity level to identify the batch in which the item or the product was produced.

## Example

```
<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 for ConfirmationRequest documents

When working with `ConfirmationRequest` documents, be aware of possible item substitutions, how confirmations orders for cancel and change orders are handled, and how order confirmations are updated.

### Item substitutions in ConfirmationRequest documents

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 service orders

`ConfirmationRequest` documents have the following limitations for service orders that require a service sheet:

- Only top-level items from the order can be included in the `ConfirmationRequest`. Otherwise, the document will be rejected.
- The `type` attribute of `ConfirmationHeader` must have one of the following values:
  - `accept` - accepts the order.
  - `reject` - rejects the order.
  - `detail` - updates portions of a purchase order, such as prices, quantities, delivery dates, reject portions, tax, and shipping information.
- If the `ConfirmationHeader` element has `type="detail"`, `ConfirmationItem` and `ConfirmationStatus` elements are included.
  - If the `ConfirmationItem` is for an outline item, the `type` attribute of the `ConfirmationStatus` element must have one of the following values:
    - `accept` - accepts this portion of the line item.
    - `reject` - rejects this portion of the line item.
    - `unknown` - allows an associated `OrderStatusRequest` (an order inquiry) to add a comment about an outline item.
  - If the `ConfirmationItem` is a top-level service item or material item, the `type` attribute of the `ConfirmationStatus` element works as it did before. It must have one of the following values:
    - `accept` - accepts this portion of the line item.
    - `detail` - accepts a portion of the line item with the changes detailed in the `ConfirmationStatus` element. At least one of the `UnitPrice`, `Shipping`, `Shipping`, `Tax`, or `ItemIn` elements, or the `deliveryDate` must be present.
    - `reject` - rejects this portion of the line item.
    - `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.

- unknown - resets the status to the default status. The item status displays on Ariba Network as if no confirmation had been sent.

### **i** Note

If the `type` attribute for `ConfirmationHeader` or `ConfirmationStatus` is an unsupported value, the document is rejected.

## Example

The following `ConfirmationRequest` accepts one top-level outline item from the order and rejects another.

```
<ConfirmationRequest>
  <ConfirmationHeader noticeDate="2015-05-04T10:55:32-07:00" type="detail"
    operation="new" confirmID="OC-4500018889">
    <Tax>
      <Money currency="USD">50.00</Money>
      <Description xml:lang="en-US"></Description>
    </Tax>
  </ConfirmationHeader>
  <OrderReference orderDate="2015-05-04T16:43:18-07:00" orderID="4500018889">
    <DocumentReference payloadID="cat0504-4500018889"></DocumentReference>
  </OrderReference>
  <ConfirmationItem itemType="composite" quantity="1" lineNumber="1">
    <UnitOfMeasure>SU</UnitOfMeasure>
    <ConfirmationStatus type="accept" quantity="1">
      <UnitOfMeasure>SU</UnitOfMeasure>
    </ConfirmationStatus>
  </ConfirmationItem>
  <ConfirmationItem itemType="composite" quantity="1" lineNumber="2">
    <UnitOfMeasure>SU</UnitOfMeasure>
    <ConfirmationStatus type="reject" quantity="1">
      <UnitOfMeasure>SU</UnitOfMeasure>
    </ConfirmationStatus>
  </ConfirmationItem>
</ConfirmationRequest>
```

## Confirmations and cancel orders

Suppliers must understand how Ariba Network processes order confirmations for canceled orders.

- Ariba Network rejects `ConfirmationRequest` documents for canceled 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.

---

## Confirmations and change orders

Suppliers must understand how Ariba Network processes order confirmations for change orders.

- Ariba Network does not allow suppliers to reject original orders after it receives changed orders.
- By default, Ariba Network persists `ConfirmationRequest` documents that are posted before the change order is posted, but does not automatically transfer or link the `ConfirmationRequest` to the new change order. Buyers can automatically transfer or link the `ConfirmationRequest` to the new change order by enabling the Ariba Network business rule **Retain confirmation status of unchanged line items on change orders**.
- 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.
- Buyers can configure their account to retain order confirmation information on changed purchase orders with the rule **Retain confirmation status of unchanged line items on change orders**.  
If the buyer enables the retention of confirmed order information on change orders, order confirmations for changed purchase orders generate a `ConfirmationRequest` document with the value "update" for the `Operation` attribute and the same `confirmID` attribute as the first order confirmation. The `ConfirmationHeader` includes a `DocumentReference` element that contains the `payloadID` of the previous order confirmation. This allows the buyer's external ERP system to update purchase order and order confirmation information rather than generating a second set of data.

## 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.  
Ariba Network uses the value in the `payloadID` or the `orderID` and `orderDate` attribute to match order confirmation and purchase order documents.  
If the `ConfirmationRequest` document contain only the `payloadID`, Ariba Network uses only the `payloadID` to match documents. When the `ConfirmationRequest` document contains the `payloadID`, `orderID` and `orderDate`, Ariba Network uses the value in the `orderID` and `orderDate` attributes to match documents.

When Ariba Network uses the `orderId` and `orderDate`, the timestamp in the `orderDate` must exactly match the date and time in the purchase order, else the `ConfirmationRequest` document will fail.

### **i** Note

As a best practice, SAP Ariba recommends buying organizations that reuse the purchase order number to send the `ConfirmationRequest` document with the `payloadID` or `orderId` and `orderDate`.

- For order confirmation, SAP Ariba Cloud Integration Gateway does not require the reference `payloadID`, but the cXML envelope for `payloadID` is required. For example, the supplier only needs to send `<OrderReference orderId="1234567890">` and leave the `payloadID` blank, `<DocumentReference payloadID=""/>`.

## Example ConfirmationRequest

The example `ConfirmationRequest` illustrates best practice use of using the purchase order number with the `payloadID`.

```
<?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>
    </ConfirmationRequest>
  </Request>
</cXML>
```



```

        </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>

```

## OrderStatusRequest documents

Buyers use the `OrderStatusRequest` document to ask suppliers for status reports on orders not yet fulfilled. Buyers can request information from suppliers on the status of the order, delivery date, current location of the shipped items, or any other related information regarding purchase orders previously sent by them.

The `OrderStatusRequest` document contains the following elements:

- `OrderStatusRequestHeader`
- `OrderStatusRequestItem`

## OrderStatusRequestHeader element

The `OrderStatusRequestHeader` element contains header-level information for the `OrderStatusRequest` document. It contains the reference information for the `OrderStatusRequestID` and `OrderStatusRequestDate` sent by the buyer.

This element has the following elements:

- `OrderReference` or `OrderIDInfo`
- `Contact`
- `Comments`
- `Extrinsic`

The `OrderStatusRequestHeader` element has the following attributes:

- `orderStatusRequestID`
- `orderStatusRequestDate`

## OrderStatusRequestItem element

The `OrderStatusRequestItem` element stores information regarding a specific line item. This element contains the following elements:

- `ItemReference`
- `Comments`

The `ItemReference` element contains the mandatory `lineNumber` attribute.

The `ConfirmationRequest` can include the `OrderStatusRequestReference` and `OrderStatusRequestID` as optional elements to explicitly reference the `OrderStatusRequest` associated to the `ConfirmationRequest`.

## Example OrderStatusRequest document

The example `OrderStatusRequest` is an illustration of using the `Comments` element.

```
<OrderStatusRequestHeader orderStatusRequestID="order02_08.20"
  orderStatusRequestDate="2013-02-06T16:09:26-08:00">
  <OrderReference orderID="order02_08.20" orderDate="2013-02-06T16:09:26-08:00">
    <DocumentReference payloadID="order02_08.20@cvbbuyer.com"/>
  </OrderReference>
  <Contact role="soldTo" addressID='AA20'>
    <Name xml:lang="en">Lisa Dollar</Name>
    <PostalAddress name='default'>
      <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>
      <Extrinsic name="POBox"></Extrinsic>
      <Extrinsic name="houseNumber"></Extrinsic>
      <Extrinsic name="building"></Extrinsic>
    </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>
  <Contact role='from' addressID='0030105956'>
    <Name xml:lang="en">Lisa Dollar</Name>
    <PostalAddress name='default'>
      <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>
<Comments>Is there any news about our order?</Comments>
</OrderStatusRequestHeader>
```

The following example shows an `OrderStatusRequestItem` element:

```
<OrderStatusRequestItem>
  <ItemReference lineNumber''10'>
    <ItemID>
      <SupplierPartID>1023</SupplierPartID>
    </itemID>
  </itemReference>
  <Comments>We did not receive the requested quantity.</Comments>
</OrderStatusRequestItem>
```

## 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.

### Related Information

[Visibility of order status \[page 194\]](#)

## ShipNoticeRequest implementation hints and limitations

The elements and attributes of `ShipNoticeRequest` have various requirements and, in some cases, limitations.

### In this section:

[ShipNoticeRequest payloadID attribute \[page 208\]](#)

[DocumentReference payloadID attribute \[page 208\]](#)

[shipmentID attribute \[page 209\]](#)

[ServiceLevel element \[page 209\]](#)

[ShipControl element \[page 209\]](#)

---

[PackageIdentification element \[page 210\]](#)  
[Packaging element \[page 210\]](#)  
[ShipNoticeItem element \[page 210\]](#)  
[shipNoticeLineNumber attribute \[page 210\]](#)  
[TransportInformation element \[page 210\]](#)  
[SupplierBatchID element \[page 210\]](#)  
[TermsofDelivery element \[page 211\]](#)  
[Ship notices and cancel and change orders \[page 211\]](#)  
[ShipNoticeHeader extrinsics \[page 211\]](#)  
[Ship notice packaging dimensions \[page 212\]](#)  
[Serial numbers and asset tags in ship notices \[page 212\]](#)  
[Limitations on ShipNoticeRequest \[page 212\]](#)

## ShipNoticeRequest payloadID attribute

The `payloadID` in the header of the `ShipNoticeRequest` must be unique.

Refer to the *cXML User's Guide* at <http://www.cxml.org> for `payloadID` format.

## DocumentReference payloadID attribute

Like the `ConfirmationRequest` transaction, the `DocumentReference` `payloadID` must match the `payloadID` of an `OrderRequest`. Ariba Network uses the value in the `payloadID` or the `orderID` and `orderDate` attribute to match documents.

If the `ShipNoticeRequest` document contain only the `payloadID`, Ariba Network uses only the `payloadID` to match documents. When the `ShipNoticeRequest` document contains the `payloadID`, `orderID` and `orderDate`, Ariba Network uses the value in the `orderID` and `orderDate` attributes to match documents.

When the `orderID` and `orderDate` attribute is used to match documents, a tolerance of +1 or -1 day is considered for the `orderDate`.

### **i** Note

As a best practice, SAP Ariba recommends buying organizations that reuse the purchase order number to send the `ShipNoticeRequest` document with the `payloadID` or `orderID` and `orderDate`.

The `orderID` is an optional attribute.

### Note

For ship notice request, SAP Ariba Cloud Integration Gateway does not require the reference `payloadID`, but the cXML envelope for `payloadID` is required. For example, the supplier only needs to send `<OrderID="1234567890">` and leave the `payloadID` blank, `<DocumentReference payloadID=""/>`.

## 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 and the tracking URL of the shipment.

## PackageIdentification element

The `PackageIdentification` 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

The `Packaging` 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 `ShipNoticeItem` stores the quantity and line number of the line item.

## shipNoticeLineNumber attribute

The `shipNoticeLineNumber` attribute is included in the `ShipNoticeItem` element and stores the ship notice line numbers.

## TransportInformation element

For more information, see [TransportInformation element \[page 131\]](#).

## SupplierBatchID element

For more information, see [ConfirmationStatus element \[page 199\]](#).

## TermsofDelivery element

You can add the `TermsofDelivery` element to the `ShipNoticeItem` element to specify terms of delivery at the line-item level.

## 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.

## ShipNoticeHeader extrinsics

`ShipNoticeHeader` has several extrinsics that can be used to specify information about packages and cargo.

`ShipNoticeHeader` has the following cXML extrinsics:

Extrinsic	Description
<code>isDivisibleLoad</code>	Indicates whether the cargo can be separated into units of legal weight without affecting the physical integrity of the load.
<code>totalOfPackages</code>	Calculated total number of packages in the shipment.
<code>isSensitiveLoad</code>	Indicates whether any of the cargo is perishable or temperature-sensitive.

The following example shows `ShipNoticeHeader` extrinsics:

```
<ShipNoticeHeader noticeDate="2017-05-17T11:17:52-07:00"
  operation="new"
  shipmentID="asn-2345678">
  <Contact role="shipFrom">
    ...
  </Contact>
  <Contact role="shipTo">
    ...
  </Contact>
  <Hazard>
    <Classification domain="UNDG">1091</Classification>
  </Hazard>
  <Packaging>
    ...
  </Packaging>
  ...
  <Extrinsic name="isDivisibleLoad">
    Yes
  </Extrinsic>
  <Extrinsic name="isSensitiveLoad">
    Yes
  </Extrinsic>
  <Extrinsic name="totalOfPackages">
    2
  </Extrinsic>
```

```
</ShipNoticeHeader>
```

## Ship notice packaging dimensions

You can use `Dimension@type` to specify width, length, and height packaging dimensions in a ship notice.

The following example shows how to use `Dimension@type` to specify packaging dimensions in a ship notice:

```
<Packaging>
  <Dimension type="length" quantity="8">
    <UnitOfMeasure>m</UnitOfMeasure>
  </Dimension>
  <Dimension type="width" quantity="5">
    <UnitOfMeasure>m</UnitOfMeasure>
  </Dimension>
  <Dimension type="height" quantity="2">
    <UnitOfMeasure>m</UnitOfMeasure>
  </Dimension>
</Packaging>
```

## Serial numbers and asset tags in ship notices

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.

### 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>
```

## Limitations on ShipNoticeRequest

Observe 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

The example ShipNoticeRequest illustrates how various elements occur within the body of the ship notice.

```
<?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"/>
        </TermsOfDelivery>
      </ShipNoticeHeader>
    </ShipNoticeRequest>
  </Request>
</cXML>
```

```

        <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>
                <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>

```

## Canceling a ship notice using cXML

You can cancel a ship notice by making several changes to it and then posting it again.

To cancel a ship notice, make the following changes to the cXML and then post it again:

- Change the payload ID
- Change the operation to 'delete'
- Change the ship notice ID (shipmentID) by appending '\_1' to the old ship notice ID
- Add a document reference to the old ship notice

### Example

```

<ShipNoticeHeader noticeDate="2016-08-11T16:42:50+03:00" operation="delete"
shipmentID="SH-1-2_1">
<DocumentReference payloadID="1471004587132-8233598467200064073@127.0.0.1">

```

---

# Order confirmation and ship notice attachments

Suppliers sometimes clarify order confirmation and ship notices with memos, faxes, or drawings.

Suppliers can use Ariba Network or cXML to attach these files using a MIME envelope. Buyers can configure Ariba Network to include attachments when forwarding order confirmations and ship notices to Ariba Buyer 8.2 or later.

## Additional References

- *cXML User's Guide* at <http://www.cxml.org>

## 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.

### 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.

# Service sheets

A service sheet is a status update for a purchase order that contains service request line items. It provides more detailed description of services than either a `StatusUpdateRequest` or `OrderConfirmationRequest`.

Suppliers do not see Services Invoicing functionality in their accounts until they receive their first service order. When they receive their first service order, Services Invoicing is automatically enabled and they can create service sheets.

Suppliers can update purchase order status either manually through their Ariba Network accounts or by submitting cXML documents to Ariba Network.

The cXML `ServiceEntryRequest` document enables suppliers to programmatically inform buying organizations of status changes to purchase orders through Ariba Network. These transactions can include both material goods and service items that the supplier added while performing the service.

## **i** Note

AribaNetwork does not support creating `ServiceEntryRequest` documents against blanket purchase orders (`ordertype="blanket"`). It does, however, support creating service sheets against regular purchase orders (`ordertype="regular"`) with unplanned blanket lines with a defined maximum amount (they must include the `<BlanketItemDetail MaxAmount=" ">` element, with the `MaxAmount` defined).

### In this section:

[Visibility of service sheet order status \[page 218\]](#)

[Ariba Network account configuration \[page 218\]](#)

[ServiceEntryRequest documents \[page 219\]](#)

[ServiceEntryRequest status updates \[page 227\]](#)

---

## Visibility of service sheet order status

Buying organizations, requisitioners, and suppliers can view order status.

## Visibility of order status to buying organizations

Authorized procurement personnel within buying organizations can log in to Ariba Network and view order status.

## Visibility of order status to requisitioners

Requisitioners (end users) have access to order status through their procurement applications.

## Visibility of order status to 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.

## ServiceEntryRequest configuration for buying organizations

Buying organizations must first contact Ariba Network Support to have their accounts enabled to download `ServiceEntryRequest` documents.

## ServiceEntryRequest configuration for suppliers

Before sending `ServiceEntryRequest` 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 suppliers log on to their Ariba Network accounts, go to the Configuration area, and set **Preferred Method of Sending Response Documents** to cXML. They must also determine the URL for posting `ServiceEntryRequest` documents by using the Profile transaction.

Suppliers must configure their backend system to use only the Ariba Network ANID of their customer in the `ToCredentials` while sending `ConfirmationRequest` or `ShipNoticeRequest` documents and not their own ANID or the Ariba Network ANID (AN01000000001).

## Related Information

[Profile transaction \[page 63\]](#)

# ServiceEntryRequest documents

A `ServiceEntryRequest` document is one that a supplier creates at a work site to provide detailed information about services provided, including material goods used during the service. A supplier sends it in response to a service line in a purchase order that requires service sheets.

Suppliers send these documents after providing services. For large or multi-part service orders, suppliers can create multiple service sheets to describe different parts of a service order or services provided during different time periods for a single line. Ariba Network receives `ServiceEntryRequest` documents, validates them against the cXML DTDs, and applies the information in them to the affected line items.

Suppliers can set the serviced status of purchase order line items by sending cXML `ServiceEntryRequest` documents. Buying organizations can see updated line item status.

## Related Information

[Visibility of service sheet order status \[page 218\]](#)

# ServiceEntryRequest implementation hints and limitations

`ServiceEntryRequest` has some required elements and attributes, and some limitations.

### In this section:

[ServiceEntryRequest payloadID attribute \[page 220\]](#)

[DocumentReference payloadID attribute \[page 220\]](#)

[ServiceEntryID attribute \[page 221\]](#)

[PartnerContact element \[page 221\]](#)

## ServiceEntryRequest payloadID attribute

The `payloadID` in the header of the `ServiceEntryRequest` must be unique.

### Additional References

- *cXML User's Guide* at <http://www.cxml.org>

## DocumentReference payloadID attribute

Like the `ConfirmationRequest` transaction, the `DocumentReference` `payloadID` must match the `payloadID` of the `OrderRequest`. Ariba Network uses the value in the `payloadID` or the `orderID` and `orderDate` attribute to match documents.

If the `ServiceEntryRequest` document contains only the `payloadID`, Ariba Network uses only the `payloadID` to match documents. When the `ServiceEntryRequest` document contains the `payloadID`, `orderID` and `orderDate`, Ariba Network uses the value in the `orderID` and `orderDate` attributes to match documents.

When the `orderID` and `orderDate` attribute is used to match documents, a tolerance of +1 or -1 day is considered for the `orderDate`.

### Note

As a best practice, SAP Ariba recommends buying organizations that reuse the purchase order number to send the `ServiceEntryRequest` document with the `payloadID` or `orderID` and `orderDate`.

The `orderID` is an optional attribute.

Every service line `DocumentReference` in a `ServiceEntryRequest` must reference the same `OrderRequest`.



## ServiceEntryID attribute

The `ServiceEntryID` in the header of the `ServiceEntryRequest` must be unique.

## PartnerContact element

The `PartnerContact` element in the header of the `ServiceEntryRequest` defines the parties involved with the service sheet. Typically it identifies the buyer's field engineer, the supplier field contractor, and the buyer approver.

- `<Contact role="fieldEngineer">` specifies the buyer field engineer who supervises the service.
- `<Contact role="fieldContractor">` specifies the supplier field contractor who provides the service.
- `<Contact role="requester">` specifies the buyer user who approves the service sheet.

You can set a service sheet transaction rule on Ariba Network to require suppliers to provide contact information for these parties in order to facilitate collaboration and resolve problems. See the *Ariba Network Buyer Administration Guide* for more information.

## Period element

The `Period` element in the header of a `ServiceEntryRequest` defines the start and end date for the service period over which the supplier performed the service. This element is optional, but if provided, it specified the default start and end dates for all line items in the service sheet.

## Automatically-generated service sheets

For any service order, suppliers can skip manual service sheet generation, directly generate service invoices for each service line, and allow the Ariba Network to automatically generate the corresponding service sheets. An automatically-generated service sheet has additional cXML elements, the same elements found on a service invoice.

The tables below list elements and attributes that have been added to automatically-generated service sheets.

Table 9: Existing elements added to the service sheet

Element	Description
PaymentTerm	Describes either the net term, the discount or penalty term in an invoice.
Tax	Taxes added to allowances and charges.
ShipNoticeIDInfo	Specifies additional reference IDs for shipment related IDs.
TotalCharges	Total sum of all the charges applied on the goods and services.

Element	Description
TotalAllowances	Total sum of all the allowances applied on the goods and services.
TotalAmountWithoutTax	Summarizes the total invoice amount without tax.
NetAmount	Total GrossAmount minus discounts.
DepositAmount	Total deposit or prepayment amount.
DueAmount	Total amount due.
SpecialHandlingAmount	Total special handling charge.
ShippingAmount	Total shipping charge.
GrossAmount	Detail-level gross payment amount.
PriceBasisQuantity	The quantity the price is quoted on.

Table 10: New elements created for the service sheet

Element	Description
ServiceEntryDetailDiscount	Mapped to InvoiceDetailDiscount
ServiceEntryItemModifications	Mapped to InvoiceItemModifications
ServiceEntryLaborDetail	Mapped to InvoiceLaborDetail
ServiceEntryDetailShipping	Mapped to InvoiceDetailShipping
ServiceEntryHeaderModifications	Mapped to InvoiceHeaderModifications
ServiceEntryDetailSummaryIndustry	Mapped to InvoiceDetailSummaryIndustry
ServiceEntryTimeCardDetail	Mapped to InvoiceTimeCardDetail
ServiceEntryDetailSummaryRetail	Mapped to InvoiceDetailSummaryRetail
ServiceEntryDetailLineIndicator	Mapped to InvoiceDetailLineIndicator

Table 11: New attributes added to ServiceEntryItem

Attribute	Description
referenceDate	The reference date for the blanket order or contract item. The usage of this attribute is optional in most cases and must be defined by the trading partners involved in the transaction. Procurement software might use this date in reconciling an invoice against a blanket order or contract.
inspectionDate	The date when the transfer of goods or the delivery of services occurs according to legal tax definitions. The usage of this attribute is optional in most cases, and must be defined by the trading partners involved in the transaction.

## Service sheets and cancel and change orders

If purchase orders are marked as Partially Serviced, Ariba Network accepts change orders for them under some circumstances.

Changes allowed for purchase orders depend on the status of the purchase orders:

Purchase order status	Changes allowed
New	Buyers can make any updates except changing a service line (requiresServiceEntry="yes") to a non-service line or vice versa. Service lines must remain service lines and non-service lines must remain non-service lines in any changes to existing orders.
Partially Serviced	<ul style="list-style-type: none"><li>Buyers cannot change service lines to non-service lines or vice versa at any time.</li><li>Buyers can add both service and material goods lines.</li><li>Buyers can edit or delete both service and material goods lines that have approved or unapproved service sheets created against them.</li></ul>

## Example ServiceEntryRequest

For reference purposes a full example of a ServiceEntryRequest is provided.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cXML.org/schemas/cXML/1.2.025/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>
    <ServiceEntryRequest>
      <ServiceEntryRequestHeader serviceEntryDate="2013-08-12T21:03:20-07:00"
        serviceEntryID="SES-10001">
        <PartnerContact>
          <Contact role="from">
            <Name xml:lang="en-US">ABC Construction</Name>
            <PostalAddress>
              <Street>125 Supplier Company Street</Street>
              <City>Sunnyvale</City>
              <State>CA</State>
              <PostalCode>94086</PostalCode>
              <Country isoCountryCode="US">United States</Country>
            </PostalAddress>
          </Contact>
        </PartnerContact>
      </ServiceEntryRequestHeader>
    </ServiceEntryRequest>
  </Request>
</cXML>
```

```

        </PostalAddress>
        <Phone>
            <TelephoneNumber>
                <CountryCode isoCountryCode="US">1</CountryCode>
                <AreaOrCityCode>800</AreaOrCityCode>
                <Number>5555555</Number>
            </TelephoneNumber>
        </Phone>
    </Contact>
</PartnerContact>
<PartnerContact>
    <Contact role="to">
        <Name xml:lang="en-US">XYZ Incorporated</Name>
        <PostalAddress>
            <Street>865 Buyer Company Street</Street>
            <Street>Suite 234</Street>
            <City>San Francisco</City>
            <State>CA</State>
            <PostalCode>94114</PostalCode>
            <Country isoCountryCode="US">United States</Country>
        </PostalAddress>
        <Phone>
            <TelephoneNumber>
                <CountryCode isoCountryCode="US">1</CountryCode>
                <AreaOrCityCode>800</AreaOrCityCode>
                <Number>5555555</Number>
            </TelephoneNumber>
        </Phone>
    </Contact>
</PartnerContact>
<PartnerContact>
    <Contact role="fieldEngineer">
        <Name xml:lang="en-US">Earl</Name>
        <Email>earl@b.com</Email>
    </Contact>
</PartnerContact>
<PartnerContact>
    <Contact role="fieldContractor">
        <Name xml:lang="en-US">Conrad</Name>
        <Email>conrad@s.com</Email>
    </Contact>
</PartnerContact>
<PartnerContact>
    <Contact role="requester">
        <Name xml:lang="en-US">Bob</Name>
        <Email>bob@b.com</Email>
    </Contact>
</PartnerContact>
<Period endDate="20013-08-09T00:00:00-07:00"
    startDate="2013-08-05T00:00:00-7:00"></Period>
</ServiceEntryRequestHeader>
<ServiceEntryOrder>
    <ServiceEntryOrderInfo>
        <OrderReference orderID="PO-HousingConstruction12345">
            <DocumentReference
                payloadID="ServiceOrderPayloadID12345">
            </DocumentReference>
        </OrderReference>
    </ServiceEntryOrderInfo>
    <ServiceEntryItem quantity="1" serviceLineNumber="1" type="service">
        <ItemReference lineNumber="1">
            <ItemID>
                <SupplierPartID>LW001-1</SupplierPartID>
            </ItemID>
            <Description xml:lang="en-US">Mobilization and
                Establish site</Description>
        </ItemReference>
        <SubtotalAmount>

```

```

        <Money currency="USD">10000.00</Money>
    </SubtotalAmount>
</ServiceEntryItem>
<ServiceEntryItem quantity="100" serviceLineNumber="2"
    type="service">
    <ItemReference lineNumber="1">
        <ItemID>
            <SupplierPartID>Cranel</SupplierPartID>
        </ItemID>
        <Description xml:lang="en-US">Service crane</Description>
    </ItemReference>
    <UnitOfMeasure>HUR</UnitOfMeasure>
    <UnitPrice>
        <Money currency="USD">500.00</Money>
    </UnitPrice>
    <SubtotalAmount>
        <Money currency="USD">50000.00</Money>
    </SubtotalAmount>
</ServiceEntryItem>
<ServiceEntryItem quantity="1000" serviceLineNumber="3"
    type="material">
    <ItemReference lineNumber="1">
        <ItemID>
            <SupplierPartID>SafetyFence</SupplierPartID>
        </ItemID>
        <Description xml:lang="en-US">Safety Fence</Description>
    </ItemReference>
    <UnitOfMeasure>FT</UnitOfMeasure>
    <UnitPrice>
        <Money currency="USD">1.00</Money>
    </UnitPrice>
    <SubtotalAmount>
        <Money currency="USD">1000.00</Money>
    </SubtotalAmount>
</ServiceEntryItem>
<ServiceEntryItem quantity="1" serviceLineNumber="4" type="service">
    <ItemReference lineNumber="2">
        <ItemID>
            <SupplierPartID>LW001-1</SupplierPartID>
        </ItemID>
        <Description xml:lang="en-US">Mobilization and
            Establish site</Description>
    </ItemReference>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <UnitPrice>
        <Money currency="USD">10000.00</Money>
    </UnitPrice>
    <SubtotalAmount>
        <Money currency="USD">10000.00</Money>
    </SubtotalAmount>
</ServiceEntryItem>
</ServiceEntryOrder>
<ServiceEntrySummary>
    <SubtotalAmount>
        <Money currency="USD">71000.00</Money>
    </SubtotalAmount>
</ServiceEntrySummary>
</ServiceEntryRequest>
</Request>
</cXML>

```

---

## Service sheet attachments

Suppliers can optionally clarify service sheets with associated memos, faxes, or drawings.

Suppliers 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 service entry sheets to their ERP systems. Ariba Network supports service sheet attachments in the `ServiceEntryHeader` only.

## 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 `ServiceEntryHeader` to indicate the existence of an attachment.

### 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.

## ServiceEntryRequest status updates

Buying organizations send `StatusUpdateRequest` documents to set service sheet status to approved or rejected. cXML-enabled suppliers can receive these documents.

The following table describes `ServiceEntryRequest` status settings:

Status	Description
approved	The buyer's ERP system has approved the service sheet.
rejected	The buyer's ERP system has rejected the service sheet, either because of validation or other errors or because a user in the approval flow denied the service sheet.

Ariba Network posts these documents to the URL that suppliers return in their cXML `ProfileResponse`. Suppliers cannot cancel service sheets at any time and they cannot invoice against rejected service sheets. They can edit and resubmit rejected service sheets.

`StatusUpdateRequest` documents identify service sheets using the `DocumentReference` element with `documentType="ServiceEntryRequest"` for the `DocumentInfo` element.

### Example

```
<StatusUpdateRequest>
  <DocumentReference payloadID="ServiceOrderPayloadID12345">
  </DocumentReference>
  <Status code="200" text="OK"></Status>
  <DocumentStatus type="approved">
    <DocumentInfo documentID="SES-HC-t1" documentType="ServiceEntryRequest"
      documentDate="2013-08-12T21:03:20-07:00">
    </DocumentInfo>
    <Comments>This service sheet has been approved.</Comments>
  </DocumentStatus>
</StatusUpdateRequest>
```

---

# Requests for quotation

Buyers send requests for quotation to SAP Ariba Sourcing either from their ERP systems or from their SAP Ariba Buying solutions. The process of sending and processing these requests for quotation varies based on the originating system. Buyers can generate RFQs in the ERP systems and use the quote automation feature to process and publish the RFQs in SAP Ariba Discovery.

## In this section:

[Quote automation \[page 228\]](#)

[Sourcing integration with SAP Ariba Buying solutions \[page 240\]](#)

## Quote automation

Buyers send the `quoteRequest` documents from their ERP system to SAP Ariba Discovery through Ariba Network. Ariba Network determines if the supplier specified in the `quoteRequest` document is a registered supplier on Ariba Network.

If the supplier is not a registered supplier, Ariba Network creates a new supplier account and sends the `quoteRequest` document to SAP Ariba Discovery to create an RFQ posting. The supplier information is also sent to SAP Ariba Discovery.

SAP Ariba Discovery matches suppliers based on the commodities and sales territories specified in the `quoteRequest` document and suppliers are invited to participate and respond to the RFQ posting. If the supplier is not registered, SAP Ariba Discovery sends an invitation requesting the supplier to register and participate in the RFQ posting.

SAP Ariba Discovery then sends all, lead or the winning bids based on the value specified in the `quoteRequest` document to the buyer's ERP system.

## Related Information

[Using request for quotations \[page 59\]](#)



---

## quoteRequest document

The `quoteRequest` document contains the `QuoteRequest` element.

The `QuoteRequest` element has the following elements:

- `QuoteRequestHeader`
- `QuoteItemOut`

The `QuoteRequestHeader` has the following optional elements:

- `Name`
- `SupplierSelector`
- `Total`
- `Description`
- `ShipTo`
- `Contact`
- `Comments`
- `Extrinsic`

## SupplierSelector element

The optional `SupplierSelector` element specifies how suppliers are selected and matched by SAP Ariba Discovery.

This element has the following attribute:

**MatchingType:** SAP Ariba Discovery uses this attribute value to determine how to match suppliers and invite them to participate in the RFQ posting on SAP Ariba Discovery. This is a mandatory field.

The `MatchingType` attribute can have the following values:

- **InvitationOnly:** Suppliers that are invited to join the RFQ posting are specified in the `OrganizationID` element.
- **approvedVendorOnly:** Suppliers that exist on Ariba Network. The supplier should be an approved supplier with the buyer. However, SAP Ariba Discovery may filter the suppliers that can bid based on the commodity and territory matching rules.
- **public:** Any public supplier. The supplier must exist on Ariba Network. However, SAP Ariba Discovery may filter the suppliers that can bid based on the commodity and territory matching rules.

The `SupplierSelector` element wraps the following elements:

- `SupplierInvitation`
- `OrganizationID`
- `Correspondent`
- `MasterAgreementIDInfo`
- `Extrinsic`

---

## SupplierInvitation

SAP Ariba Discovery uses the `SupplierInvitation` element to identify the supplier that is invited to join the RFQ posting.

The `SupplierInvitation` element has the following attribute:

- `supplierStatus`

The `supplierStatus` attribute can have the following values:

- `approved`: The supplier exists in the buyer's system and is approved by the buyer. The default value is "approved."
- `contracted`: The supplier is an approved supplier in the buyer's system and has an associated master agreement of a contract. The buyer can specify the `MasterAgreementIDInfo`.

## OrganizationID

The `OrganizationID` element is a unique identification for the supplier on Ariba Network. This element is used by the buyer to specify suppliers that are invited to bid.

## Correspondent

The optional `Correspondent` element stores the contact information of the supplier and is used to identify and contact the supplier.

If the supplier does not exist, Ariba Network uses the contact information to send an invitation to the supplier to register on Ariba Network

---

## MasterAgreementIDInfo

The optional `MasterAgreementIDInfo` element is the ID number of the buyer for the corresponding master agreement of the contract or release order. This element is enhanced with the `IdReference` element.

## Extrinsic

The optional `Extrinsic` element allows buyers to provide additional information for the `SupplierInvitation`. Buyers can specify more than one `Extrinsic` element.

## processingTarget extrinsic

Quote automation allows you to configure the flow of RFQs, whether you want Ariba Network to route them to SAP Ariba Discovery or SAP Ariba Sourcing.

By default, the routing configuration is set to SAP Ariba Sourcing which means that Ariba Network routes the flow of RFQ events to SAP Ariba Sourcing. However, if the RFQ is originating from SAP S/4HANA ERP system, you can control the flow of each RFQ by specifying the following extrinsic in the `quoteRequest` document:

### processingTarget

This extrinsic specified in the header of the `quoteRequest` cXML document overrides the RFQ routing configuration set in Ariba Network for quote automation. You can specify one of the following values in the **processingTarget** extrinsic:

- **QuoteAutomation**
- **sourcingRequest**

An example of the code snippet containing the extrinsic:

```
<Extrinsic name="processingTarget">sourcingRequest</Extrinsic>
```

If you specify the value of the extrinsic as **QuoteAutomation**, Ariba Network processes the RFQ event and routes it to SAP Ariba Discovery irrespective of the routing configuration set in Ariba Network.

## quoteMessage document

The `quoteMessage` document contain a `QuoteMessage` element.

The `QuoteMessage` element has the following elements:

- `QuoteMessageHeader`
- `QuoteItemIn`

For more information on the `quoteRequest` and `quoteMessage` documents, see the *cXML User's Guide*.

## Example quoteRequest document

A buyer sends a quoteRequest document to supplier to invite them to bid. The following example shows a sample of the quoteRequest document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/schemas/cXML/1.2.025/Quote.dtd">
<cXML payloadID="123" timestamp="2013-11-19T11:50:11+00:00" version="1.2.025">
<Header>
  <From>
    <Credential domain="NetworkId">
      <Identity>AN02000533043</Identity>
    </Credential>
  </From>
  <To>
    <Credential domain="NetworkId">
      <Identity>AN02000533043</Identity>
    </Credential>
  </To>
  <Sender>
    <Credential domain="NetworkID">
      <Identity>AN02000533043</Identity>
      <SharedSecret>sharedsecret</SharedSecret>
    </Credential>
    <UserAgent>Procurement App 2.0</UserAgent>
  </Sender>
</Header>
<Request deploymentMode="production">
  <QuoteRequest>
    <QuoteRequestHeader requestID="2021.2215"
      requestDate="2014-03-06T23:25:01.000-07:00"
      type="new" openDate="2014-03-10T23:13:49.000-07:00"
      currency="USD" xml:lang="en"
      closeDate="2014-03-17T00:00:00.000-07:00"
      quoteReceivingPreference="winningOnly">
      <SupplierSelector matchingType="invitationOnly">
        <SupplierInvitation>
          <OrganizationID>
            <Credential domain="VendorID">
              <Identity>2210</Identity>
            </Credential>
          </OrganizationID>
          <Correspondent>
            <Contact>
              <Name xml:lang="en">New Inv Inc</Name>
              <PostalAddress>
                <DeliverTo>922 West</DeliverTo>
                <Street>15th Avenue</Street>
                <City>California</City>
                <State>CA</State>
                <PostalCode>93504</PostalCode>
                <Country isoCountryCode="US"></Country>
              </PostalAddress>
              <Email>vp@abc.com</Email>
              <Phone>
                <TelephoneNumber>
                  <CountryCode isoCountryCode="US"></CountryCode>
                  <AreaOrCityCode>ca</AreaOrCityCode>
                  <Number>2342344</Number>
                </TelephoneNumber>
              </Phone>
              <URL></URL>
            </Contact>
          </Correspondent>
        </SupplierInvitation>
      </SupplierSelector>
    </QuoteRequestHeader>
  </QuoteRequest>
</Request>
</cXML>
```

```

    <Description xml:lang="" >Quote for AS1</Description>
    <ShipTo>
      <Address isoCountryCode="US" addressID="409" >
        <Name xml:lang="en" >P2- Los Angeles</Name>
        <PostalAddress name="P2- Los Angeles" >
          <Street>1022 Sepulveda Blvd</Street>
          <Street>Suite 1000Address2</Street>
          <Street>Address3</Street>
          <City>El Segundo</City>
          <State>California</State>
          <PostalCode>90245</PostalCode>
          <Country isoCountryCode="US" >United States</Country>
        </PostalAddress>
        <Phone>
          <TelephoneNumber>
            <CountryCode isoCountryCode="US" />
            <AreaOrCityCode/>
            <Number/>
          </TelephoneNumber>
        </Phone>
        <Fax>
          <TelephoneNumber>
            <CountryCode isoCountryCode="US" />
            <AreaOrCityCode/>
            <Number/>
          </TelephoneNumber>
        </Fax>
      </Address>
    </ShipTo>
  </QuoteRequestHeader>
  .
  .
  .

```

## quoteMessage Document

The quoteMessage document describes the items or services on which the buyer is requesting quotes. The following example shows a sample of the quoteMessage document:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/cXML/1.2.025/Quote.dtd">
<cXML payloadID="5565217623@10.10.14.16" timestamp="2013-08-11T08:50:12-07:00"
version="1.2.025" xml:lang="en-US">
  <Header>
    <From>
      <Credential domain="NetworkID">
        <Identity>AN02000873997</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="NetworkID">
        <Identity>AN70000000073</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="NetworkID">
        <Identity>AN01000874827</Identity>
        <SharedSecret>sharedsecret</SharedSecret>
      </Credential>
      <UserAgent>Procurement App 2.0</UserAgent>
    </Sender>
  </Header>
  <Message>

```

```

<QuoteMessage>
  <QuoteMessageHeader xml:lang="en" currency="USD"
    quoteDate="2013-10-03T08:24:15-07:00"
    quoteID="T123" type="award">
    <OrganizationID>
      <Credential domain="NetworkID">
        <Identity>AN02000873997</Identity>
      </Credential>
    </OrganizationID>
    <Total>
      <Money currency="USD">235400.0</Money>
    </Total>
    <QuoteRequestReference requestDate="2013-10-03T08:14:54-07:00"
      requestID="2021.2215"/>
  </QuoteMessageHeader>
  <QuoteItemIn rank="1" lineNumber="1" quantity="100" type="award">
    <ItemID>
      <SupplierPartID>SupplierPartID-1</SupplierPartID>
      <SupplierPartAuxiliaryID>SupplierPartAuxID-1
      </SupplierPartAuxiliaryID>
      <BuyerPartID/>
    </ItemID>
    <ItemDetail>
      <UnitPrice>
        <Money currency="USD">950.00</Money>
      </UnitPrice>
      <Description xml:lang="en">Laptops</Description>
      <UnitOfMeasure>EA</UnitOfMeasure>
      <Classification domain="unspsc">43211503</Classification>
      <ManufacturerPartID>ManufacturerPartID-1</ManufacturerPartID>
      <ManufacturerName>ManufacturerName-1</ManufacturerName>
      <URL>http://www.abc.com-1</URL>
      <LeadTime>31</LeadTime>
    </ItemDetail>
    <Shipping>
      <Money currency="USD">100.0000000000</Money>
      <Description xml:lang="en">Maximum shipping
        allowance-1</Description>
    </Shipping>
    <Tax>
      <Money currency="USD">100.0000000000</Money>
      <Description xml:lang="en">Maximum tax allowance-1</Description>
    </Tax>
    <Total>
      <Money currency="USD">95200.00</Money>
    </Total>
  </QuoteItemIn>
  <QuoteItemIn rank="1" lineNumber="2" quantity="200" type="award">
    <ItemID>
      <SupplierPartID>SupplierPartID-2</SupplierPartID>
      <SupplierPartAuxiliaryID>SupplierPartAuxID-2
      </SupplierPartAuxiliaryID>
      <BuyerPartID/>
    </ItemID>
    <ItemDetail>
      <UnitPrice>
        <Money currency="USD">700.00</Money>
      </UnitPrice>
      <Description xml:lang="en">Desktop Computers</Description>
      <UnitOfMeasure>EA</UnitOfMeasure>
      <Classification domain="unspsc">43211507</Classification>
      <ManufacturerPartID>ManufacturerPartID-2</ManufacturerPartID>
      <ManufacturerName>ManufacturerName-2</ManufacturerName>
      <URL>http://www.abc.com-2</URL>
      <LeadTime>32</LeadTime>
    </ItemDetail>
    <Shipping>
      <Money currency="USD">100.0000000000</Money>

```

```

        <Description xml:lang="en">Maximum shipping
            allowance-2</Description>
    </Shipping>
    <Tax>
        <Money currency="USD">100.0000000000</Money>
        <Description xml:lang="en">Maximum tax allowance-2</Description>
    </Tax>
    <Total>
        <Money currency="USD">140200.00</Money>
    </Total>
</QuoteItemIn>
</QuoteMessage>
</Message>

```

## Canceling requests for quotation

Buying organizations can cancel a request for quotation that was sent to the sourcing application. Ariba Network displays the status of canceled requests for quotation as "Obsoleted."

Canceled requests for quotation are the same as regular requests for quotation, except the `QuoteRequestHeader` element has the attribute `type="delete"` instead of "new", and the `requestID` attribute of the original request for quotation.

The following example shows how to cancel a request for quotation:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "http://xml.cxml.org/schemas/schemas/cXML/1.2.029/Quote.dtd">
<cXML payloadID="123" timestamp="2016-02-02T11:50:11+00:00" version="1.2.029">
    <Header>
        <From>
            <Credential domain="NetworkId">
                <Identity>AN02000533043</Identity>
            </Credential>
        </From>
        <To>
            <Credential domain="NetworkId">
                <Identity>AN02000533043</Identity>
            </Credential>
        </To>
        <Sender>
            <Credential domain="NetworkID">
                <Identity>AN02000533043</Identity>
                <SharedSecret>sharedsecret</SharedSecret>
            </Credential>
            <UserAgent>Procurement App 2.0</UserAgent>
        </Sender>
    </Header>
    <Request deploymentMode="production">
        <QuoteRequest>
            <QuoteRequestHeader
                requestID="6000000131"
                requestDate="2016-02-02T13:56:00-5:00"
                type="delete"
                openDate="2016-02-02T13:56:00-5:00"
                closeDate="2016-03-02T13:56:00-5:00"
                currency="USD"
                xml:lang="en"
                quoteReceivingPreference="winningOnly">
                ...
            </QuoteRequestHeader>
        </QuoteRequest>
    </Request>

```

```
</cXML>
```

## SAP Ariba Sourcing integration with SAP ERP

Several cXML extrinsic elements accommodate fields that are sent from SAP ERP Central Component (ECC) to SAP Ariba Sourcing. SAP sends a `QuoteRequest` document to SAP Ariba Sourcing with information that SAP Ariba Sourcing uses to create a sourcing request. In most cases, the sourcing request is used to create a sourcing event that includes data and items from the sourcing request. After the event ends, SAP Ariba Sourcing can send award or pricing information back to SAP in a `QuoteMessage` document.

### Header level extrinsics

Header level extrinsics are sent as child elements of the `QuoteHeaderInfo` element.

- `CompanyCode.Id`
- `CompanyCode.Name`
- `PurchasingOrganization.Id`
- `PurchasingOrganization.Name`
- `PurchasingGroup.Id`
- `PurchasingGroup.Name`
- `PaymentTerms.Id`
- `PaymentTerms.Name`

### Line item level extrinsics

Line item level extrinsics are sent in `ItemDetail` elements contained in `QuoteItemOut` and `QuoteItemIn` elements.

- `Plant.Id`
- `Plant.Name`
- `MaterialGroup.Id`
- `MaterialGroup.Name`
- `ItemCategory`
- `MaterialCode.Id`
- `MaterialCode.Name`
- `Incoterms.Id`
- `Incoterms.Name`
- `RequisitionId`
- `RFQId`
- `RequisitionLineItemNumber`



- RFQLineItemNumber

## Example: quoteMessage extrinsics for integration with SAP ERP

The following example shows both header level and line item level extrinsics used in a quoteMessage document for integration with SAP ERP.

```
<QuoteMessage>
  <QuoteMessageHeader currency="USD" quoteDate="2015-03-12T01:05:53-08:00"
    quoteID="QA170" type="award" xml:lang="en">
    <OrganizationID>
      <Credential domain="NetworkID">
        <Identity>AN1000000123</Identity>
      </Credential>
    </OrganizationID>
    <Total>
      <Money currency="USD">5000.00</Money>
    </Total>
    <QuoteRequestReference requestDate="2015-03-12T00:42:06-08:00"
      requestID="6000000174"/>
    <Extrinsic name="Terms">
      <Extrinsic name="CompanyCode.Id">3000</Extrinsic>
      <Extrinsic name="CompanyCode.Name">BestRun USA</Extrinsic>
      <Extrinsic name="PurchasingOrganization.Id">3000</Extrinsic>
      <Extrinsic name="PurchasingOrganization.Name">PurchOrg US</Extrinsic>
      <Extrinsic name="PurchasingGroup.Id">001</Extrinsic>
      <Extrinsic name="PurchasingGroup.Name">Dickens, B.</Extrinsic>
      <Extrinsic name="PaymentTerms.Id">0001</Extrinsic>
      <Extrinsic name="PaymentTerms.Name">Quick Pay</Extrinsic>
      <Extrinsic name="DocumentType">PurchaseOrder</Extrinsic>
      <Extrinsic name="DocumentCategory">NB</Extrinsic>
    </Extrinsic>
  </QuoteMessageHeader>
  <QuoteItemIn lineNumber="10" quantity="500" rank="1"
    requestedDeliveryDate="2015-03-27T03:55:00-07:00" type="award">
    <ItemID>
      <SupplierPartID/>
      <SupplierPartAuxiliaryID/>
      <BuyerPartID/>
    </ItemID>
    <ItemDetail>
      <UnitPrice>
        <Money currency="USD">10.00</Money>
      </UnitPrice>
      <Description xml:lang="en">ERP-10810</Description>
      <UnitOfMeasure>BX</UnitOfMeasure>
      <Classification domain="unspsc">43211503</Classification>
      <ManufacturerPartID/>
      <ManufacturerName/>
      <URL/>
      <LeadTime/>
      <Extrinsic name="Terms">
        <Extrinsic name="Plant.Id">3000</Extrinsic>
        <Extrinsic name="Plant.Name">New York</Extrinsic>
        <Extrinsic name="ItemCategory">0</Extrinsic>
        <Extrinsic name="MaterialGroup.Id">00801</Extrinsic>
        <Extrinsic name="MaterialGroup.Name">Disposable paper</Extrinsic>
        <Extrinsic name="Incoterms.Id">CFR</Extrinsic>
        <Extrinsic name="Incoterms.Name">Incoterms for RFQ</Extrinsic>
        <Extrinsic name="MaterialCode.Id"/>
        <Extrinsic name="MaterialCode.Name"/>
        <Extrinsic name="RequisitionId">0010017303</Extrinsic>
        <Extrinsic name="RFQId">6000000174</Extrinsic>
      </Extrinsic>
    </ItemDetail>
  </QuoteItemIn>
</QuoteMessage>
```

```

    <Extrinsic name="RequisitionLineItemNumber">00010</Extrinsic>
    <Extrinsic name="RFQLineItemNumber">00010</Extrinsic>
  </Extrinsic>
</ItemDetail>
<Total>
  <Money currency="USD">5000.00</Money>
</Total>
</QuoteItemIn>
</QuoteMessage>

```

## SAP service items and service hierarchies (cXML adapter version 1.2.034 or later)

A `quoteRequest` or `quoteMessage` document can contain data for SAP service items (SAP Item Category D items) in a service hierarchy.

`quoteRequest` documents received from SAP and `quoteMessage` documents sent to SAP have the following items in `QuoteItemOut` and `QuoteItemIn` elements for service hierarchies:

- The attribute `itemClassification` with the value `service`.
- The `itemType` attribute, set to `composite` for service lines and service outlines, or set to `item` for service specifications.
- The `parentLineNumber` and `lineNumber` attributes are used to link parent and child items, such as a parent service line and child service specification. The value of a child's `parentLineNumber` attribute matches its parent's `lineNumber` attribute.
- The child element `Classification` has the attribute `code` set to `Services` and the attribute `domain` set to `MaterialGroup`. The value of the `Classification` element is an SAP material group.

The following is an excerpt of a `quoteRequest` document with 3 `QuoteItemOut` elements: a service line, service outline, and service specification.

```

<QuoteItemOut itemClassification="service" itemType="composite" quantity="1.000"
requestedDeliveryDate="2017-08-28T07:45:50+05:30" lineNumber="00010">
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD"/>
    </UnitPrice>
    <Description xml:lang="en">Services</Description>
    <UnitOfMeasure>SU</UnitOfMeasure>
    <Classification code="Services" domain="MaterialGroup">007</Classification>
  </ItemDetail>
  <ReferenceDocumentInfo lineNumber="00010">
    <DocumentInfo documentType="Requisition" documentID="0010021706">Constant</
DocumentInfo>
  </ReferenceDocumentInfo>
  <Comments xml:lang="en">true </Comments>
</QuoteItemOut>
<QuoteItemOut itemClassification="service" itemType="composite" quantity="0.000" parentLineNumber="00010"
lineNumber="0000200010">
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD"/>
    </UnitPrice>
    <Description xml:lang="en">Set up Construction site</Description>
    <UnitOfMeasure></UnitOfMeasure>
    <Classification domain="MaterialGroup"></Classification>
  </ItemDetail>
  <ReferenceDocumentInfo lineNumber="00010">
    <DocumentInfo documentType="RFQ" documentID="6000001310"/>
  </ReferenceDocumentInfo>
</QuoteItemOut>

```

```

    </ReferenceDocumentInfo>
  </QuoteItemOut>
  <QuoteItemOut itemType="item" quantity="1.000" parentLineNumber="0000200010"
  serviceLineType="standard" lineNumber="1000300010">
    <ItemID>
      <SupplierPartID/>
      <BuyerPartID>0000000000000100000</BuyerPartID>
    </ItemID>
    <ItemDetail>
      <UnitPrice>
        <Money currency="USD"/>
      </UnitPrice>
      <Description xml:lang="en">Set up construction site</Description>
      <UnitOfMeasure>SU</UnitOfMeasure>
      <Classification domain="MaterialGroup">007</Classification>
    </ItemDetail>
    <ReferenceDocumentInfo lineNumber="0000300010">
      <DocumentInfo documentType="RFQ" documentID="6000001310"/>
    </ReferenceDocumentInfo>
  </QuoteItemOut>

```

### SAP simple service items (cXML adapter version 1.2.031, 1.2.032, or 1.2.033)

A `quoteRequest` or `quoteMessage` document can contain data for SAP service items (SAP Item Category D items) that have a single-level hierarchy.

When SAP sends a service specification to SAP Ariba, a parent `QuoteItemOut` element and a child `QuoteItemOut` element are sent; SAP Ariba combines data from the two `QuoteItemOut` elements to create a single line item in the SAP Ariba sourcing request.

A child `QuoteItemOut` element is linked to a parent `QuoteItemOut` element using the `parentLineNumber` extrinsic; the `QuoteItemOut` element with the matching `lineNumber` attribute value is the parent.

If the SAP RFQ contains a service specifications outline with multiple service specifications, one parent `QuoteItemOut` element is sent with multiple child `QuoteItemOut` elements (one `QuoteItemOut` element for each service specification).

When SAP Ariba sends award information in a `quoteMessage` to SAP, it sends a parent `QuoteItemIn` and child `QuoteItemIn` pair for each service line item.

The parent `QuoteItemOut` or `QuoteItemIn` element for a service item includes an `ItemDetail` element with an `ItemCategory` extrinsic that has the value `service`.

The `ItemDetail` element in `QuoteItemOut` and `QuoteItemIn` elements for service items also includes a `Classification` element. The `Classification` element's `domain` attribute and value are mapped to an SAP Ariba Strategic Sourcing solutions commodity code domain and value.

# Sourcing integration with SAP Ariba Buying solutions

Buyers send the `QuoteRequest` document, also known as a sourcing request, from SAP Ariba Buying solutions to SAP Ariba Sourcing. Currently, buyers can send sourcing requests only for aggregated requisitions when using an integrated solution including SAP Ariba Sourcing.

The `quoteRequest` document is sent directly from SAP Ariba Buying solutions to SAP Ariba Sourcing. A sourcing agent creates a sourcing project for the sourcing request in SAP Ariba Sourcing and invites suppliers to participate in the sourcing event. After the suppliers have submitted their bids, the sourcing agent awards the bid to the most suitable suppliers, generates the pricing terms, and sends the pricing terms back to SAP Ariba Buying solutions, where they are applied on the requisition. The pricing terms are sent using the `QuoteDataMessage`.

## QuoteRequest documents sent by SAP Ariba Buying solutions

When a purchasing agent creates a sourcing request for an aggregated requisition in SAP Ariba Buying solutions, a sourcing request cXML file, also known as `QuoteRequest` document, is sent to SAP Ariba Sourcing. This file contains details of the line items in the aggregated requisition.

For sourcing requests sent by SAP Ariba Buying solutions, the `Identity` element within the `Sender` element in the `Header` holds the value `Buyer:` followed by the Ariba Network ID of the site sending the sourcing request. The details of each line item in the aggregated requisition are sent using a `QuoteItemOut` element in the `QuoteRequest` element.

## Sample QuoteRequest document sent by SAP Ariba Buying solutions

The sample `QuoteRequest` document given below contains the details of two line items having descriptions `Aircraft` and `AirPart`.

### Sample Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "%cxmlSource%">
<cXML timestamp="2014-12-04T14:59:46+05:30"
payloadID="1417685386200.809200779.000000002@M0gq09EOjMo5PAOF3eZazxOPhjY="
xml:lang="en-US" version="1.2.023">

  <Header>
    <!-- class: ariba.encoder.xml.AXComponent -->
    <From>
      <!-- class: ariba.encoder.xml.AXComponent -->
      <Credential domain="NetworkId">
        <!-- class: ariba.encoder.xml.AXComponent -->
        <Identity>Buyer:AN02001100452</Identity>
      </Credential>
```

```

</From>
<To>
  <!-- class: ariba.encoder.xml.AXComponent -->
  <Credential domain="NetworkId">
    <!-- class: ariba.encoder.xml.AXComponent -->
    <Identity>ACM:AN02001100452</Identity>
  </Credential>
</To>
<Sender>
  <Credential domain="NetworkId">
    <!-- class: ariba.encoder.xml.AXComponent -->
    <Identity>Buyer:AN02001100452</Identity>
    <SharedSecret>***shared secret removed***</SharedSecret>
  </Credential>
  <UserAgent>Buyer 13s25</UserAgent>
</Sender>
</Header>
<Request>
  <QuoteRequest>
    <QuoteRequestHeader quoteReceivingPreference="winningOnly" type="New"
requestDate="Wed Dec 03 17:17:32 IST 2014" requestID="PR18">
    </QuoteRequestHeader>
    <!-- class:
ariba.purchasing.sourcing.RequisitionSourcingRequestEncodeBodyContent -->
    <QuoteItemOut quantity="5.0000000000" lineNumber="1">
      <ItemDetail>
        <UnitPrice>
          <!-- class: ariba.procure.server.cxml.MoneyComponent -->
          <Money alternateCurrency="" alternateAmount="" currency="USD">100</
Money>
        </UnitPrice>
        <Description xml:lang="en">Aircraft</Description>
        <UnitOfMeasure>5 each</UnitOfMeasure>
      </ItemDetail>
    </QuoteItemOut>
    <QuoteItemOut quantity="4.0000000000" lineNumber="2">
      <ItemDetail>
        <UnitPrice>
          <!-- class: ariba.procure.server.cxml.MoneyComponent -->
          <Money alternateCurrency="" alternateAmount="" currency="USD">10</
Money>
        </UnitPrice>
        <Description xml:lang="en">AirPart</Description>
        <UnitOfMeasure>4 each</UnitOfMeasure>
      </ItemDetail>
    </QuoteItemOut>
  </QuoteRequest>
</Request>
</cXML>

```

## QuoteDataMessage documents sent from SAP Ariba Sourcing

After the sourcing project is processed and the bid has been awarded to suppliers, the sourcing agent generates and sends the resulting pricing terms from SAP Ariba Sourcing to SAP Ariba Buying solutions in a cXML file called as a `QuoteDataMessage` document.

A single cXML file contains the pricing terms for all the line items in the aggregated requisition. The pricing terms for each supplier are sent using a `QuoteData` element. The supplier's ID is mentioned in the `Identity` element within

the `OrganizationID` element. The pricing terms for each line item are mentioned in the `QuoteItemIn` element. The `Extrinsic` element contains additional details about the pricing terms, such as charges and discounts.

## Sample QuoteDataMessage document sent to SAP Ariba Buying solutions

The sample pricing terms file given below contains pricing terms for two line items `Nc1` and `Nc2`. Both have been awarded to the same supplier with buyer system ID `SID515`.

### Sample Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM "%CXMLDTDURL%">
<!-- class: ariba.cxml.Component -->
<cXML timestamp="2015-11-12T20:04:44-08:00" payloadID="%payload-id%" xml:lang="en-US" version="1.2.028">
  <Header>
    <!-- class: ariba.encoder.xml.AXComponent -->
    <From>
      <!-- class: ariba.encoder.xml.AXComponent -->
      <Credential domain="SpendManagementNetwork">
        <!-- class: ariba.encoder.xml.AXComponent -->
        <Identity>ACM:AN02001565298</Identity>
      </Credential>
    </From>
    <To>
      <!-- class: ariba.encoder.xml.AXComponent -->
      <Credential domain="SpendManagementNetwork">
        <!-- class: ariba.encoder.xml.AXComponent -->
        <Identity>Buyer:AN02001565298</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="SpendManagementNetwork">
        <!-- class: ariba.encoder.xml.AXComponent -->
        <Identity>ACM:AN02001565298</Identity>
        <SharedSecret>dummy</SharedSecret>
      </Credential>
      <UserAgent>ACM 14s1</UserAgent>
    </Sender>
  </Header>
  <!-- class: ariba.sourcing.integration.networkrfq.QuoteDataMessageEncode -->
  <Message>
    <QuoteDataMessage>
      <QuoteDataMessageHeader>
        <QuoteRequestReference requestDate="2015-11-12T19:34:37-08:00" requestID="%req-id%"/>
      </QuoteDataMessageHeader>
      <!-- class: ariba.sourcing.integration.networkrfq.QuoteMessageEncode -->
      <QuoteData>
        <QuoteDataInfo xml:lang="en" currency="USD" quoteDate="2015-11-12T20:04:07-08:00" quoteID="ID105918" type="award">
          <OrganizationID>
            <Credential domain="buyersystemid">
              <Identity>SID515</Identity>
            </Credential>
          </OrganizationID>
          <Total>
            <Money currency="USD">174.25</Money>
          </Total>
        </QuoteDataInfo>
      </QuoteData>
    </QuoteDataMessage>
  </Message>
</cXML>
```

```

<!-- class: ariba.sourcing.integration.networkrfq.QuoteItemInEncode -->
<QuoteItemIn rank="1" lineNumber="%line-item%" quantity="1" type="award">
  <ItemID>
    <SupplierPartID>RANDOM123</SupplierPartID>
    <SupplierPartAuxiliaryID></SupplierPartAuxiliaryID>
    <BuyerPartID></BuyerPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">95.00</Money>
    </UnitPrice>
    <Description xml:lang="en">Nc1</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="unspsc">2513</Classification>
    <ManufacturerPartID></ManufacturerPartID>
    <ManufacturerName></ManufacturerName>
    <URL></URL>
    <LeadTime>3</LeadTime>
    <Extrinsic name="Terms">
      <!-- class:
ariba.sourcing.integration.networkrfq.QuoteItemExtrinsicEncode -->
      <Extrinsic name="DISCOUNTAMT"><Money currency="USD">5.00</Money></
Extrinsic>
      <!-- class:
ariba.sourcing.integration.networkrfq.QuoteItemExtrinsicEncode -->
      <Extrinsic name="DISCOUNTPCT">5.00</Extrinsic>
      <!-- class:
ariba.sourcing.integration.networkrfq.QuoteItemExtrinsicEncode -->
      <Extrinsic name="SURCHARGEAMT"><Money currency="USD">5.00</Money></
Extrinsic>
      <!-- class:
ariba.sourcing.integration.networkrfq.QuoteItemExtrinsicEncode -->
      <Extrinsic name="SURCHARGEPCPT">5.00</Extrinsic>
    </Extrinsic>
  </ItemDetail>
  <Total>
    <Money currency="USD">94.76</Money>
  </Total>
</QuoteItemIn>
<!-- class: ariba.sourcing.integration.networkrfq.QuoteItemInEncode -->
<QuoteItemIn rank="1" lineNumber="%line-item%" quantity="1" type="award">
  <ItemID>
    <SupplierPartID>RANDOM123</SupplierPartID>
    <SupplierPartAuxiliaryID></SupplierPartAuxiliaryID>
    <BuyerPartID></BuyerPartID>
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">80.00</Money>
    </UnitPrice>
    <Description xml:lang="en">Nc2</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="unspsc">2513</Classification>
    <ManufacturerPartID></ManufacturerPartID>
    <ManufacturerName></ManufacturerName>
    <URL></URL>
    <LeadTime>0</LeadTime>
    <Extrinsic name="Terms">
      <!-- class:
ariba.sourcing.integration.networkrfq.QuoteItemExtrinsicEncode -->
      <Extrinsic name="DISCOUNTAMT"><Money currency="USD">8.00</Money></
Extrinsic>
      <!-- class:
ariba.sourcing.integration.networkrfq.QuoteItemExtrinsicEncode -->
      <Extrinsic name="DISCOUNTPCT">8.00</Extrinsic>
      <!-- class:
ariba.sourcing.integration.networkrfq.QuoteItemExtrinsicEncode -->

```

```

        <Extrinsic name="SURCHARGEAMT"><Money currency="USD">8.00</Money></
Extrinsic>
        <!-- class:
ariba.sourcing.integration.networkrfq.QuoteItemExtrinsicEncode -->
        <Extrinsic name="SURCHARGEPCCT">8.00</Extrinsic>
        </Extrinsic>
    </ItemDetail>
    <Total>
        <Money currency="USD">79.49</Money>
    </Total>
</QuoteItemIn>
</QuoteData>
</QuoteDataMessage>
</Message>
</cXML>

```



---

# Invoices and scheduled payments

Suppliers generate invoices manually, through cXML `InvoiceDetailRequests`, or EDI documents. Buying organizations use scheduled payments (`PaymentProposalRequest` documents) to tell Ariba Network and their suppliers about planned payments.

## In this section:

- [Basic invoice functionality \[page 245\]](#)
- [Purchase order matching \[page 246\]](#)
- [Matching in Ariba Buyer \[page 247\]](#)
- [Invoicing business rules \[page 247\]](#)
- [Invoice implementation hints \[page 249\]](#)
- [Transmission of invoices to buying organizations \[page 303\]](#)
- [Status and cancel invoices \[page 303\]](#)
- [Example summary invoice \[page 304\]](#)
- [Example header-level invoice \[page 308\]](#)
- [Example blanket purchase order invoice \[page 311\]](#)
- [Example of a complex buyer/supplier scenario \[page 313\]](#)
- [Scheduled payments \[page 325\]](#)

## Basic invoice functionality

Suppliers can generate invoices manually using the online invoice generator (PO-Flip), through cXML `InvoiceDetailRequest` documents, or through EDI ANSI X12 810 or EDIFACT INVOIC documents. Suppliers can use any of these methods without making configuration changes on Ariba Network, but 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.

Buyers must enable the rule for suppliers to create PO-based and non-PO invoices. 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 documents might reside on Ariba Network, or they might be external documents (those documents 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 \[page 247\]](#).)

---

## 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 \[page 247\]](#).

## 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.

Examples of business rules include:

- Allow invoices that contain detailed service information for both goods and service items but that are not based on service sheets. 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 non-service 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.

### Note

SAP Ariba Cloud Integration Gateway receives the invoices and then posts it to the Ariba Network. If the invoice is transmitted successfully, the **Transaction Tracker** in the cloud integration gateway will show it as successful. Ariba Network then sends a `StatusUpdate` to the cloud integration gateway for the invoice against the buyers business rules. The status of Invoice in transaction tracker will not be updated but you can map the `StatusUpdate` received by SAP Ariba Cloud Integration Gateway to suppliers acknowledgement formats.

For any other document type, when the document is posted by SAP Ariba Cloud Integration Gateway, Ariba Network responds synchronously and the status is reflected in the **Transaction Tracker** against the transaction.

## Rules that affect PO-flip only

Some business rules only affect manually-created online purchase orders (PO-Flip), but not invoices generated through cXML, EDI, or CSV.

The following business rules affect only PO-Flip invoices:

- 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.

---

# Invoice implementation hints

SAP Ariba applications have specific requirements for invoice contents.

## In this section:

- [eSignatures \[page 250\]](#)
- [Supplier self-signed invoices \[page 250\]](#)
- [IdReference element \[page 253\]](#)
- [Contact roles \[page 254\]](#)
- [Conversion of Segment to AccountingSegment \[page 255\]](#)
- [Invoice payloadID attribute \[page 255\]](#)
- [ServiceEntryItemReference element \[page 256\]](#)
- [SerialNumber elements \[page 256\]](#)
- [PaymentTerm element \[page 257\]](#)
- [Remittance IDs \[page 257\]](#)
- [Credit memos and line-item credit memos \[page 257\]](#)
- [Inspection date attribute \[page 258\]](#)
- [PriceBasisQuantity element \[page 258\]](#)
- [itemType attribute \[page 259\]](#)
- [compositeltemType attribute \[page 260\]](#)
- [Tolerances element \[page 262\]](#)
- [TaxDetail category \[page 263\]](#)
- [Tax on shipping and special handling \[page 263\]](#)
- [Allowances and charges \[page 265\]](#)
- [French parafiscal taxes \[page 273\]](#)
- [Withholding tax support \[page 278\]](#)
- [Mandatory fields for countries with a VAT system \[page 279\]](#)
- [Invoices for suppliers from Mexico \[page 285\]](#)
- [Attachments with invoices \[page 289\]](#)
- [Extrinsic elements \[page 290\]](#)
- [Invoice element field lengths \[page 301\]](#)
- [PDF invoice copy attachments \[page 302\]](#)

## 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 either country requires 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)
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 **Review Invoice** 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 [https://knowledge.ariba.com/Ariba\\_eSigning\\_CountryMatrix](https://knowledge.ariba.com/Ariba_eSigning_CountryMatrix).

## Supplier self-signed invoices

Ariba Network supports self-signed invoices. Suppliers who implement their own digital signature solution (via cXML) can create self-signed invoices.

The self-signing 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

## eSigning limitations

Buyers should fully understand the supplier's process for eSigning and validating on their behalf. If the invoice is not canonicalized before signing, any third party (for example, Infomosaic, Apache) validation of the signature fails. SAP 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`.

### 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 Network guide to invoice conversion*.

## Other invoice 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`.

## 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`.

### 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.
- `departmentName`—Identifies the Japanese address and honorific format on the Ariba Network. In addition, allows to enter the name of the person or organization to whom the service is being delivered.
- `supplierReference`—Internal supplier reference ID used to identify outbound documents to the supplier user.

## ACH information

ACH information provides details on where to send payments.

### 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="accountID" />
  <IdReference identifier="Savings" domain="accountType" />
  <IdReference identifier="Sunnyvale" domain="branchName" />
</InvoicePartner>
```

## Wire information

Wire information provides details on where to send payments.

### 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="accountID" />
  <IdReference identifier="Savings" domain="accountType" />
  <IdReference identifier="Sunnyvale" domain="branchName" />
</InvoicePartner>
```

## Contact roles

Invoices can provide `Contact` roles that define a supplier's name and address, and the name and address of the buyer.

`Contact` roles can have the following 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

The `Segment` element was deprecated and replaced by the `AccountingSegment` element in cXML 1.2.005.

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.

## 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.

Invoices should also contain the `payloadID` of the corresponding purchase order or master agreement. Use the guidelines for formulating `payloadID` in the *cXML User's Guide*.

### **i** Note

For invoices, SAP Ariba Cloud Integration Gateway does not require the reference `payloadID`, but the cXML envelope for `payloadID` is required. For example, the supplier only needs to send `<OrderID="1234567890">` and leave the `payloadID` blank, `<DocumentReference payloadID=""/>`.

## Related Information

[Purchase order matching \[page 246\]](#)

## ServiceEntryItemReference element

The `ServiceEntryItemReference` element references the service sheet line item on which the invoice line item is based.

`ServiceEntryItemReference` has the following attributes:

A new element that references the originating service sheet. It has the following attributes:

- `serviceEntryDate`, the date when the service sheet was created
- `serviceEntryID`, the unique identifier for the service sheet
- `serviceLineNumber`, the service sheet line number

It includes a `<DocumentReference>` element that references the service sheet itself. For example:

```
<InvoiceDetailItem invoiceLineNumber="1" quantity="1">
  <UnitOfMeasure>EA</UnitOfMeasure>
  <UnitPrice>
    <Money currency="USD">9000</Money>
  </UnitPrice>
  <InvoiceDetailItemReference lineNumber="1">
    <ItemID>
      <SupplierPartID>cat1010m</SupplierPartID>
    </ItemID>
    <Description xml:lang="en-US"></Description>
  </InvoiceDetailItemReference>
  <ServiceEntryItemReference serviceEntryDate="2013-08-12T21:03:20-07:00"
    ServiceEntryID="SES-HC-t1" serviceLineNumber="1">
    <DocumentReference
      payloadID="1381430779525-5274812674141413083@10.10.15.142">
    </DocumentReference>
  </ServiceEntryItemReference>
  <SubtotalAmount>
    <Money currency="USD">9000</Money>
  </SubtotalAmount>
  <GrossAmount>
    <Money currency="USD">9000</Money>
  </GrossAmount>
  <NetAmount>
    <Money currency="USD">9000</Money>
  </NetAmount>
</InvoiceDetailItem>
```

## 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.

Buying organizations can also require the presence of payment terms on invoices.

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.

### 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.

## Credit memos and line-item credit memos

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.

## Example

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/>
    <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` attribute set to the `payloadID` of the original invoice ( `<DocumentReference payloadID="PreviousInvoicePayloadId"/>`). For example, `<DocumentReference payloadID="5555555555@10.10.1.1"/>`. 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.

## 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

The `PriceBasisQuantity` element contains the quantity-based pricing for a line item.

For more information, see [PriceBasisQuantity element \[page 150\]](#).

This element is also available in invoices with `InvoiceDetailServiceItem` elements as long as they are not based on underlying service sheets. Invoices created from service sheets do not support `PriceBasisQuantity` 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.

### 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>
```

```
</InvoiceDetailItemReference>
</InvoiceDetailItem>
```

## compositeItemType attribute

The `compositeItemType` is used to support item groups in non-PO invoices.

- The `compositeItemType` attribute is used for DTD tag `Item.mod`. Its possible values are `groupLevel` and `itemLevel`.
- The `compositeItemType` entity is added as an optional attribute to the following elements:
  - `ItemIn`
  - `ItemOut`
  - `InvoiceDetailItem`
  - `ConfirmationItem`
  - `ShipNoticeItem`
  - `ReceiptItem`

The following example shows an item group with group-level pricing type:

```
<InvoiceDetailOrder>
  <InvoiceDetailOrderInfo>
    <OrderIDInfo orderID=""></OrderIDInfo>
  </InvoiceDetailOrderInfo>
  <InvoiceDetailItem quantity="1" invoiceLineNumber="1"
    itemType="composite" compositeItemType="groupLevel">
    <UnitOfMeasure></UnitOfMeasure>
    <UnitPrice>
      <Money currency="USD">21.00</Money>
    </UnitPrice>
    <InvoiceDetailItemReference lineNumber="1">
      <ItemID>
        <SupplierPartID>1</SupplierPartID>
      </ItemID>
      <Description xml:lang="en">Parent Item</Description>
    </InvoiceDetailItemReference>
    <TotalAllowances>
      <Money currency="USD">25.00</Money>
    </TotalAllowances>
    <TotalAmountWithoutTax>
      <Money currency="USD">290.00</Money>
    </TotalAmountWithoutTax>
    <NetAmount>
      <Money currency="USD">290.00</Money>
    </NetAmount>
  </InvoiceDetailItem>
  <InvoiceDetailItem invoiceLineNumber="2" quantity="15"
    parentInvoiceLineNumber="1" itemType="item">
    <UnitOfMeasure>33</UnitOfMeasure>
    <UnitPrice>
      <Money currency="USD">21.00</Money>
    </UnitPrice>
    <InvoiceDetailItemReference lineNumber="1">
      <ItemID>
        <SupplierPartID>1</SupplierPartID>
      </ItemID>
      <Description xml:lang="en">Child Item</Description>
    </InvoiceDetailItemReference>
    <SubtotalAmount>
```



```

    <Money currency="USD">315.00</Money>
  </SubtotalAmount>
  <GrossAmount>
    <Money currency="USD">290.00</Money>
  </GrossAmount>
  <InvoiceItemModifications>
  <Modification>
    <AdditionalDeduction>
      <DeductionAmount>
        <Money currency="USD">47.25</Money>
      </DeductionAmount>
      <DeductionPercent percent="15"></DeductionPercent>
    </AdditionalDeduction>
    <ModificationDetail name="Contract Allowance">
      <Description xml:lang="en"/>
    </ModificationDetail>
  </Modification>
</InvoiceItemModifications>
  <TotalAllowances>
    <Money currency="USD">25.00</Money>
  </TotalAllowances>
  <TotalAmountWithoutTax>
    <Money currency="USD">290.00</Money>
  </TotalAmountWithoutTax>
  <NetAmount>
    <Money currency="USD">290.00</Money>
  </NetAmount>
</InvoiceDetailItem>
</InvoiceDetailOrder>

```

The following example shows an item group with item-level pricing type:

```

<InvoiceDetailOrder>
  <InvoiceDetailOrderInfo>
    <OrderIDInfo orderID=""></OrderIDInfo>
  </InvoiceDetailOrderInfo>
  <InvoiceDetailItem quantity="1" invoiceLineNumber="1"
  itemType="composite" compositeltemType="itemLevel">
    <UnitOfMeasure></UnitOfMeasure>
    <UnitPrice>
      <Money currency="USD">0.00</Money>
    </UnitPrice>
    <InvoiceDetailItemReference lineNumber="1">
      <ItemID>
        <SupplierPartID>1</SupplierPartID>
      </ItemID>
      <Description xml:lang="en">Parent Item</Description>
    </InvoiceDetailItemReference>
  </InvoiceDetailItem>
  <InvoiceDetailItem invoiceLineNumber="2" quantity="15"
  parentInvoiceLineNumber="1" itemType="item">
    <UnitOfMeasure>33</UnitOfMeasure>
    <UnitPrice>
      <Money currency="USD">21.00</Money>
    </UnitPrice>
    <InvoiceDetailItemReference lineNumber="1">
      <ItemID>
        <SupplierPartID>1</SupplierPartID>
      </ItemID>
      <Description xml:lang="en">Child Item</Description>
    </InvoiceDetailItemReference>
  <SubtotalAmount>
    <Money currency="USD">315.00</Money>
  </SubtotalAmount>
  <GrossAmount>
    <Money currency="USD">290.00</Money>
  </GrossAmount>

```

```

    <InvoiceItemModifications>
    <Modification>
    <AdditionalDeduction>
    <DeductionAmount>
    <Money currency="USD">47.25</Money>
    </DeductionAmount>
    <DeductionPercent percent="15" />
    </AdditionalDeduction>
    <ModificationDetail name="Contract Allowance">
    <Description xml:lang="en"></Description>
    </ModificationDetail>
    </Modification>
  </InvoiceItemModifications>
  <TotalAllowances>
    <Money currency="USD">25.00</Money>
  </TotalAllowances>
  <TotalAmountWithoutTax>
    <Money currency="USD">290.00</Money>
  </TotalAmountWithoutTax>
  <NetAmount>
    <Money currency="USD">290.00</Money>
  </NetAmount>
</InvoiceDetailItem>
</InvoiceDetailOrder>

```

## Tolerances element

When a purchase order contains the line item quantity and unit price tolerances, these tolerances are applied to invoices created on Ariba Network.

Ariba Network applies the line item quantity and unit price tolerances to invoices in the `InvoiceDetailRequest` cXML documents sent through cXML. The line item quantity and unit price tolerances from the purchase order does not appear in invoices also. However, when suppliers create an invoice, Ariba Network validates the line item quantity tolerance and unit price tolerance does not exceed the specified tolerance limit in the purchase order.

When suppliers create an invoice for a changed purchase order, the line item quantity and unit price tolerances available in the changed purchase order is used to validate the line items in the invoice. The tolerances specified in the original purchase order are ignored.

Ariba Network applies the line item quantity and unit price tolerances in invoices as the following:

- When the buyer enables the following rules:  
**Allow suppliers to increase item quantities**  
**Allow suppliers to increase item quantities on ship notices**  
**Allow suppliers to create invoices based on received quantities only OR Allow suppliers to create invoices based on shipped quantities only**

Case 1:

When the shipped or received line item quantity is less than the purchase order line item quantity, and the purchase order has the line item quantity tolerance, Ariba Network allows suppliers to invoice only for the shipped or received line item quantity.

For example,

You send a purchase order with the following:

Quantity for line item 1 = 100.

Line item quantity tolerance for line item 1 = 10%. Total quantity for line item 1 that can be shipped or received is 110.

Quantity shipped or received for line item 1 = 50.

Ariba Network allows suppliers to invoice line item 1 for a quantity of 50 only.

Case 2:

When the shipped or received line item quantity is more than the purchase order line item quantity, and the purchase order has the line item quantity tolerance, Ariba Network allows suppliers to invoice only for the line item quantity that does not exceed the specified tolerance limit in the purchase order.

For example,

You send a purchase order with the following:

Quantity for line item 1 = 100.

Line item quantity tolerance for line item 1 = 10%. Total quantity for line item 1 that can be shipped or received is 110.

Quantity shipped or received for line item 1 = 150.

Ariba Network allows suppliers to invoice line item 1 for a quantity of 110 only.

For more information, see [Tolerances element \[page 151\]](#).

## TaxDetail 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:

TaxDetail category attribute	Description
sales	Sales Tax
vat	Value Added Tax
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.

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
    ====
  -->
</InvoiceDetailItem>
```

```

        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>

```

## Allowances and charges

Ariba Network allows suppliers to include additional charges, allowances, and their taxes that are incurred for the total landed cost of the goods and services for line items on an invoice.

## Shipping and SpecialHandling

Details of header-level shipping and special handling charges are sent using the `ShippingAmount` and `SpecialHandlingAmount` elements while details of line-level shipping and special handling charges are sent using the `InvoiceDetailLineShipping` and `InvoiceDetailLineSpecialHandling` elements.

In the sample code given below, a special handling charge of USD 10 has been applied on the line item.

### Sample Code

```

<InvoiceDetailOrder>
    <InvoiceDetailItem>
        <InvoiceDetailLineSpecialHandling>
            <Description xml:lang="en-US">Special handling</Description>
            <Money currency="USD">10.00</Money>
        </InvoiceDetailLineSpecialHandling>
    </InvoiceDetailItem>
</InvoiceDetailOrder>
<InvoiceDetailSummary>

```

```

    <SpecialHandlingAmount>
      <Money currency="USD">10.00</Money>
    </SpecialHandlingAmount>
    <TotalCharges>
      <Money currency="USD">10.00</Money>
    </TotalCharges>
  </InvoiceDetailSummary>

```

## Discount

Details of discounts applied on the invoice are sent using the `InvoiceDetailDiscount` element at the header and line level.

```

<InvoiceDetailOrder>
  <InvoiceDetailItem>
    <InvoiceDetailDiscount>
      <Money currency="USD">15.00</Money>
    </InvoiceDetailDiscount>
  </InvoiceDetailItem>
</InvoiceDetailOrder>
<InvoiceDetailSummary>
  <InvoiceDetailDiscount>
    <Money currency="USD">15.00</Money>
  </InvoiceDetailDiscount>
</InvoiceDetailSummary>

```

## level Attribute

The `Modification` element has an optional attribute `level`, which can have an integer value. It is used for compound allowances and charges and it displays on which level a certain modification is applied.

The following example shows how the `level` attribute is used for an item with several modifications:

```

<InvoiceHeaderModifications>
  <Modification level="1">
    <OriginalPrice>5.500</OriginalPrice>
    <AdditionalDeduction>
      <DeductionPercent>2</DeductionPercent>
    </AdditionalDeduction>
  </Modification>
  <Modification level="1">
    <OriginalPrice>5.500</OriginalPrice>
    <AdditionalCost>
      <Money currency="USD">2.00</Money>
    </AdditionalCost>
  </Modification>
  <Modification level="2">
    <OriginalPrice 7.390</OriginalPrice >
    <AdditionalDeduction>
      <DeductionPercent>10</DeductionPercent>
    </AdditionalDeduction>
  </Modification>
</InvoiceHeaderModifications>

```

## InvoiceHeaderModifications

The `InvoiceHeaderModifications` element specifies the additional charges, allowances, and their taxes that are incurred for the total landed cost of the goods and services at the invoice header-level.

The `InvoiceHeaderModifications` element can store one or more [Modification](#) elements [page 136].

The `AdditionalDeduction` element with `type="withholdingTax"` contains withholding taxes. For each line item in a `PaymentRemittanceRequest`, the value displayed in the `Money` element contained by the `AdjustmentAmount` element is the sum of the values displayed in the `Money` elements contained by the `DeductionAmount` elements, which are contained by `AdditionalDeduction` elements with the attributes `type="withholdingTax"` and `type="other"`.

In the `PaymentRemittanceSummary`, the value displayed in the `Money` element contained by the `AdjustmentAmount` element is the sum of all adjustment amounts in the remittance.

The following example shows `AdjustmentAmount` and `Comments` elements from the `InvoiceHeaderModifications`:

```
<AdjustmentAmount>
  <Money currency="USD">110.00</Money>
  <Modifications>
    <Modification>
      <AdditionalDeduction type="withholdingTax">
        <DeductionAmount>
          <Money currency="USD">95.00</Money>
        </DeductionAmount>
      </AdditionalDeduction>
    </Modification>
    <Modification>
      <AdditionalDeduction type="other">
        <DeductionAmount>
          <Money currency="USD">15.00</Money>
        </DeductionAmount>
      </AdditionalDeduction>
    </Modification>
  </Modifications>
</AdjustmentAmount>
<Comments>Tax Withheld</Comments>
```

## InvoiceItemModifications

The `InvoiceItemModifications` element specifies the additional charges, allowances, and their taxes for the total landed cost of the goods and service for an invoice line-item.

The `InvoiceItemModifications` element can store one or more `Modification` elements. For more information on the `Modification` element, see [Modifications element](#) [page 136].

---

## TotalCharges

The `TotalCharges` element is the total sum of all the charges applied on the goods and services. It can appear at the line-item and summary in an invoice.

## TotalAllowances

The `TotalAllowances` element is the total sum of all the allowances applied on the goods and services. It can appear at the line item and summary in an invoice.

## TotalAmountWithoutTax

The `TotalAmountWithoutTax` element is used to summarize the total invoice amount without tax.

The total amount includes:

- SubTotal
- Shipping Amount
- Special Handling
- Charges

Allowances and discounts are subtracted from the sum of the above four amounts.

The following examples specify the allowances, charges, and their taxes in an invoice. The first example displays line-item allowances and charges at the line level and the second displays unit price modifications.

- Modifications at the line level:

```
<InvoiceDetailItem invoiceLineNumber = "1" quantity = "1">
  <UnitOfMeasure>ea</UnitOfMeasure>
  <UnitPrice>
    <Money currency = "USD">1000.00</Money>
  </UnitPrice>
  <InvoiceDetailItemReference lineNumber = "1">
    <ItemID>
      <SupplierPartID>part#1</SupplierPartID>
    </ItemID>
    <Description xml:lang = "en-US"></Description>
  </InvoiceDetailItemReference>
  <SubtotalAmount>
    <Money currency = "USD">1000.00</Money>
  </SubtotalAmount>
  <GrossAmount>
    <Money currency = "USD">1000.00</Money>
  </GrossAmount>
  <InvoiceItemModifications>
    <Modification>
      <AdditionalCost>
        <Money currency = "USD">13</Money>
      </ AdditionalCost>
      <ModificationDetail name = "Insurance"
        startDate = "2013-09-06T10:15:00-08:00"
        endDate = "2013-09-10T10:15:00-08:00"/>
    </Modification>
  </InvoiceItemModifications>
</InvoiceDetailItem>
```



```

</InvoiceItemModifications>
<TotalCharges>
  <Money currency="USD">13</Money>
</TotalCharges>
<TotalAmountWithoutTax>
  <Money currency="USD">1013</Money>
</TotalAmountWithoutTax>
<NetAmount>
  <Money currency = "USD">1013.00</Money>
</NetAmount>
</InvoiceDetailItem>
</InvoiceDetailOrder>
<InvoiceDetailSummary>
  <SubtotalAmount>
    <Money currency = "USD">1000.00</Money>
  </SubtotalAmount>
  <Tax>
    <Money currency = "USD">10.0</Money>
    <Description xml:lang = "en-US"></Description>
    <TaxDetail category="TaxOnAllowance" percentageRate = "10">
      <TaxableAmount>
        <Money currency = "USD">100.00</Money>
      </TaxableAmount>
      <TaxAmount>
        <Money currency = "USD">10.00</Money>
      </TaxAmount>
      <Description xml:lang = "en-US"></Description>
    </TaxDetail>
  </Tax>
  <GrossAmount>
    <Money currency = "USD">1000.00</Money>
  </GrossAmount>
  <InvoiceHeaderModifications>
    <Money currency="USD">100</Money>
    <Modification>
      <AdditionalDeduction>
        <Money currency = "USD">100</Money>
      </AdditionalDeduction>
      <Tax>
        <Money currency = "USD">10.00</Money>
        <Description xml:lang = "en-US"></Description>
        <TaxDetail category="TaxOnAllowance" percentageRate="10">
          <TaxableAmount>
            <Money currency = "USD">100.00</Money>
          </TaxableAmount>
          <TaxAmount>
            <Money currency = "USD">10.00</Money>
          </TaxAmount>
          <Description xml:lang = "en-US"></Description>
        </TaxDetail>
      </Tax>
      <ModificationDetail name = "SpecialDiscount"
        startDate = "2013-09-06T10:15:00-08:00"
        endDate = "2013-09-10T10:15:00-08:00"/>
    </Modification>
  </InvoiceHeaderModifications>
  <TotalCharges>
    <Money currency="USD">13</Money>
  </ TotalCharges>
  <TotalAllowances>
    <Money currency="USD">100</Money>
  </ TotalAllowances>
  <TotalAmountWithoutTax>
    <Money currency = "AUD">913.00</Money>
  </TotalAmountWithoutTax>
  <NetAmount>
    <Money currency = "USD">923.00</Money>
  </NetAmount>

```

```

        <DueAmount>
            <Money currency = "USD">923.00</Money>
        </DueAmount>
    </InvoiceDetailSummary>
</InvoiceDetailRequest>

```

- Modifications at unit price level:

```

<InvoiceDetailItem invoiceLineNumber="1" quantity="10">
    <UnitOfMeasure>LB</UnitOfMeasure>
    <UnitPrice>
        <Money currency="USD">15.00</Money>
        <Modifications>
            <Modification>
                <OriginalPrice>
                    <Money currency="USD">25.00</Money>
                </OriginalPrice>
                <AdditionalDeduction>
                    <DeductionAmount>
                        <Money currency="USD">10.00</Money>
                    </DeductionAmount>
                </AdditionalDeduction>
                <ModificationDetail name="Discount-Special">
                    <Description xml:lang="en">Test Discount</Description>
                </ModificationDetail>
            </Modification>
        </Modifications>
    </UnitPrice>
    <InvoiceDetailItemReference lineNumber="1">
        <ItemID>
            <SupplierPartID>1</SupplierPartID>
        </ItemID>
        <Description xml:lang="en">Test Item 1</Description>
    </InvoiceDetailItemReference>
    <SubtotalAmount>
        <Money currency="USD">150.00</Money>
    </SubtotalAmount>
    <GrossAmount>
        <Money currency="USD">150.00</Money>
    </GrossAmount>
    <TotalAmountWithoutTax>
        <Money currency="USD">150.00</Money>
    </TotalAmountWithoutTax>
    <NetAmount>
        <Money currency="USD">150.00</Money>
    </NetAmount>
</InvoiceDetailItem>
<InvoiceDetailSummary>
    <SubtotalAmount>
        <Money currency="USD">150.00</Money>
    </SubtotalAmount>
    <Tax>
        <Money currency="USD">0</Money>
        <Description xml:lang="en"/>
    </Tax>
    <GrossAmount>
        <Money currency="USD">150.00</Money>
    </GrossAmount>
    <InvoiceDetailSummaryLineItemModifications>
        <Modification>
            <AdditionalDeduction>
                <DeductionAmount>
                    <Money currency="USD">100.00</Money>
                </DeductionAmount>
            </AdditionalDeduction>
            <ModificationDetail name="Discount-Special">
                <Description xml:lang="en">Test Discount</Description>
            </ModificationDetail>
        </Modification>
    </InvoiceDetailSummaryLineItemModifications>

```

```

        </Modification>
    </InvoiceDetailSummaryLineItemModifications>
    <TotalAmountWithoutTax>
        <Money currency="USD">150.00</Money>
    </TotalAmountWithoutTax>
    <NetAmount>
        <Money currency="USD">150.00</Money>
    </NetAmount>
    <DueAmount>
        <Money currency="USD">150.00</Money>
    </DueAmount>
</InvoiceDetailSummary>

```

## Money or percentage allowances and charges

The following examples show how to add allowances and charges as percentage rate or as fixed amount:

- Modification on header level with amount charge

```

<InvoiceHeaderModifications>
    <Modification>
        <AdditionalCost>
            <Money currency="USD">25.00</Money>
        </AdditionalCost>
        <ModificationDetail name="Other charges">
            <Description xml:lang="en">Test Charge</Description>
        </ModificationDetail>
    </Modification>
</InvoiceHeaderModifications>

```

- Modification on header level with percentage charge

```

<InvoiceHeaderModifications>
    <Modification>
        <AdditionalCost>
            <Percentage percent="10"/>
        </AdditionalCost>
        <ModificationDetail name="Other charges">
            <Description xml:lang="en">Test Charge</Description>
        </ModificationDetail>
    </Modification>
</InvoiceHeaderModifications>

```

- Modification on line level with amount charge

```

<InvoiceItemModifications>
    <Modification>
        <AdditionalCost >
            <Money currency="USD">25.00</Money>
        </AdditionalCost>
        <ModificationDetail name="Other charges">
            <Description xml:lang="en">Test Charge Line</Description>
        </ModificationDetail>
    </Modification>
</InvoiceItemModifications>

```

- Modification on line level with percentage charge

```

<InvoiceItemModifications>
    <Modification>
        <AdditionalCost >
            <Percentage percent="10"/>
        </AdditionalCost>
    </Modification>
</InvoiceItemModifications>

```

```

        </AdditionalCost>
        <ModificationDetail name="Other charges">
            <Description xml:lang="en">Test Charge Line</Description>
        </ModificationDetail>
    </Modification>
</InvoiceItemModifications>

```

- Modification on header level with amount allowance

```

<InvoiceHeaderModifications>
    <Modification>
        <AdditionalDeduction>
            <DeductionAmount><Money currency="USD">5.00</Money></DeductionAmount>
        </AdditionalDeduction>
        <ModificationDetail name="Volume Discount">
            <Description xml:lang="en"> Test Discount</Description>
        </ModificationDetail>
    </Modification>
</InvoiceHeaderModifications>

```

- Modification on header level with percentage allowance

```

<InvoiceHeaderModifications>
    <Modification>
        <AdditionalDeduction>
            <DeductionPercent percent="10"/>
        </AdditionalDeduction>
        <ModificationDetail name="Volume Discount">
            <Description xml:lang="en"> Test Discount</Description>
        </ModificationDetail>
    </Modification>
</InvoiceHeaderModifications>

```

- Modification on line level with amount allowance

```

<InvoiceItemModifications>
    <Modification>
        <OriginalPrice>
            <Money currency="USD">25.00</Money>
        </OriginalPrice>
        <AdditionalDeduction>
            <DeductionAmount><Money currency="USD">5.00</Money></DeductionAmount>
        </AdditionalDeduction>
        <ModificationDetail name="Discount-Special">
            <Description xml:lang="en">Test Discount Line</Description>
        </ModificationDetail>
    </Modification>
</InvoiceItemModifications>

```

- Modification on line level with percentage allowance

```

<InvoiceItemModifications>
    <Modification>
        <OriginalPrice>
            <Money currency="USD">25.00</Money>
        </OriginalPrice>
        <AdditionalDeduction>
            <DeductionPercent percent="10"/>
        </AdditionalDeduction>
        <ModificationDetail name="Discount-Special">
            <Description xml:lang="en">Test Discount Line</Description>
        </ModificationDetail>
    </Modification>
</InvoiceItemModifications>

```

---

## ModificationDetail names

The following values can be used for the `name` attribute of `ModificationDetail`:

- Access Charge
- AccountNumberCorrectionCharge
- AcidBattery
- AdditionalPackaging
- Adjustment
- Allowance
- Carrier
- Charge
- ChargeForCommercialDiscount
- ChargeForPreferentialPositioning
- ChargesForReturnedGoods
- CollectionFee
- Contract Allowance
- CustomFees
- Discount-Special
- FinancialFees
- Freight
- FreightBasedOnDollarMinimum
- Insurance
- Handling
- Honorarium
- Labelling
- OrderCharges
- OtherCharges
- OrderOfFullPalette
- Packaging
- Parafiscal Tax
- ReturnableGoodsCharge
- Royalties
- ServiceCharges
- Volume Discount

## French parafiscal taxes

Ariba Network allows suppliers to include French parafiscal taxes on invoices. Parafiscal taxes are mandatory taxes, collected for dedicated funds, enforced by French law. Suppliers can add French parafiscal taxes by using the following elements:

- `ModificationDetail name Parafiscal Tax`. It is used for charges of type parafiscal tax.
- `Extrinsic name code`. It is used for the code of the added parafiscal tax.

The following cXML excerpts show examples of Parafiscal Tax ModificationDetail with the available parafiscal taxes:

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Droit eaux et boissons non alcool </Description>
  <Extrinsic name="code">3001000002008</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Droit spécif or argent et platine</Description>
  <Extrinsic name="code">3001000002022</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe forestière</Description>
  <Extrinsic name="code">3001000002039</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe CODIFAB</Description>
  <Extrinsic name="code">3001000002046</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe DEFI</Description>
  <Extrinsic name="code">3001000002060</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Redevance édition et ouvr librairie</Description>
  <Extrinsic name="code">3001000002077</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe HBJOAT</Description>
  <Extrinsic name="code">3001000002084</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Surtaxe sur les eaux minérales</Description>
  <Extrinsic name="code">3001000002091</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe financement organisme agricole</Description>
  <Extrinsic name="code">3001000002107</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Redevance sanitaire d'abattage</Description>
  <Extrinsic name="code">3001000002114</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe municipale sur l'électricité</Description>
  <Extrinsic name="code">3001000002169</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe département sur l'électricité</Description>
  <Extrinsic name="code">3001000002176</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Redevance sanitaire de découpage</Description>
  <Extrinsic name="code">3001000002183</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Cotisation Interbev</Description>
  <Extrinsic name="code">3001000002190</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe CTIFL</Description>
  <Extrinsic name="code">3001000002206</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Octroi de mer</Description>
  <Extrinsic name="code">3001000002213</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">TICGN</Description>
  <Extrinsic name="code">3001000002237</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Cotisation Unicid</Description>
  <Extrinsic name="code">3001000002244</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">CVRO Gazon</Description>
  <Extrinsic name="code">3001000002251</Extrinsic>
</ModificationDetail>
```

```
  <Description xml:lang="fr">Taxe du CN du cinématographe</Description>
  <Extrinsic name="code">3001000002268</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe CIFOG</Description>
  <Extrinsic name="code">3001000002275</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Contribution Eco-emballages</Description>
  <Extrinsic name="code">3001000002282</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">TIPP</Description>
  <Extrinsic name="code">3001000002299</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">TGAP</Description>
  <Extrinsic name="code">3001000002305</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Droits d'accises sur les alcools</Description>
  <Extrinsic name="code">3001000002312</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Cotisation sécurité sociale</Description>
  <Extrinsic name="code">3001000002329</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Contribution Corepile</Description>
  <Extrinsic name="code">3001000002336</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Cotisation Interfel</Description>
  <Extrinsic name="code">3001000002367</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Cotisation label viande</Description>
  <Extrinsic name="code">3001000002381</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe CETIM</Description>
  <Extrinsic name="code">3001000002398</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Autre CVO interprofessionnelle</Description>
  <Extrinsic name="code">3001000002404</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Taxe d'abatage</Description>
  <Extrinsic name="code">3001000002411</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">CTA Gaz</Description>
  <Extrinsic name="code">3001000002435</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">CSPE</Description>
  <Extrinsic name="code">3001000002442</Extrinsic>
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
  <Description xml:lang="fr">Contribution DEEE</Description>
  <Extrinsic name="code">3001000002459</Extrinsic>
```



```
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">CTA Electricité</Description>  
  <Extrinsic name="code">3001000002466</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">Rémunération pour copie privée</Description>  
  <Extrinsic name="code">3001000002473</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">Contribution pour une pêche durable</Description>  
  <Extrinsic name="code">3001000002480</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">Redevance pour pollutions diffuses</Description>  
  <Extrinsic name="code">3001000002497</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">Taxe pour le dvpt ind construction</Description>  
  <Extrinsic name="code">3001000002503</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">Compensation gaz tarif solidarité</Description>  
  <Extrinsic name="code">3001000002510</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">Contribution Eco TLC</Description>  
  <Extrinsic name="code">3001000002527</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">Taxe vidéogrammes</Description>  
  <Extrinsic name="code">3001000002534</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">Taxe sur les prémix</Description>  
  <Extrinsic name="code">3001000002541</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">CSE Porc</Description>  
  <Extrinsic name="code">3001000002596</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">  
  <Description xml:lang="fr">TICFE</Description>  
  <Extrinsic name="code">3001000002619</Extrinsic>  
</ModificationDetail>
```

```
<ModificationDetail name="Parafiscal Tax">
```

```

    <Description xml:lang="fr">TDCFE</Description>
    <Extrinsic name="code">3001000002626</Extrinsic>
  </ModificationDetail>

```

```

  <ModificationDetail name="Parafiscal Tax">
    <Description xml:lang="fr">TCCFE</Description>
    <Extrinsic name="code">3001000002633</Extrinsic>
  </ModificationDetail>

```

## 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.

### Example

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>

```

---

## Mandatory 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.

### In this section:

- [Mandatory line item description \[page 279\]](#)
- [Mandatory Bill To \[page 280\]](#)
- [Mandatory Remit To \[page 280\]](#)
- [Mandatory Ship From and Ship To \[page 281\]](#)
- [Mandatory buyer VAT ID and supplier VAT ID \[page 282\]](#)
- [Mandatory supplier information \[page 282\]](#)
- [Mandatory supply date \[page 283\]](#)
- [Mandatory VAT in both buyer's and supplier's local currency \[page 284\]](#)
- [Mandatory explanation of zero-value VAT entries \[page 284\]](#)

## Mandatory line item description

Some countries may require mandatory line item descriptions.

### Example

The following example shows a line item description:

```
<InvoiceDetailItemReference lineNumber="1">
  ...
  <Description xml:lang="en">Blue Ballpoint Pens, Retractable</Description>
</InvoiceDetailItemReference>
```

## Mandatory Bill To

Some countries may require a mandatory Bill To contact.

### Example

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

Some countries may require a mandatory Remit To contact. In case the Remit To contact is a factoring service, it may also be to mark that on invoices.

### Examples

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>l</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>

```

The following example shows a Remit To contact, which is a factoring service:

```

<InvoiceDetailRequestHeader invoiceDate="2014-06-02T16:14:13+03:00"
invoiceID="IN0123"
invoiceOrigin="supplier" operation="new" purpose="standard">
    <InvoiceDetailHeaderIndicator/>
    <InvoiceDetailLineIndicator/>
    <InvoicePartner>
        <Contact role="remitTo">
            <Name xml:lang="en">Factoring Service</Name>
            <PostalAddress>
                <Street>test</Street>
                <City>Sofia</City>
                <PostalCode>1000</PostalCode>
                <Country isoCountryCode="BG">Bulgaria</Country>
            </PostalAddress>
            <Extrinsic name="isFactoring">true</Extrinsic>
        </Contact>
    </InvoicePartner>
</InvoiceDetailRequestHeader>

```

## Mandatory Ship From and Ship To

Some countries may require mandatory Ship From and Ship To fields.

### Example

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>
            <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>
            </TelephoneNumber>
        </Phone>
    </Contact>

```

```

        <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.

### Example

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.

### Note

Ariba Buyer, SAP Ariba Buying and Invoicing and SAP Ariba Invoice Management support the following maximum number of characters. Any field values exceeding the limit will be truncated to:

- `supplierCommercialIdentifier`: 50 characters
- `supplierCommercialCredentials`: 100 characters
- `legalStatus`: 50 characters
- `legalCapital`: 50 characters

## Example

```
<InvoiceDetailRequestHeader>
  ...
  <Extrinsic name="supplierCommercialIdentifier">1234567890</Extrinsic>
  <Extrinsic name="supplierCommercialCredentials">CEO</Extrinsic>
  <Extrinsic name="legalStatus">Inc.</Extrinsic>
  <Extrinsic name="legalCapital">1000</Extrinsic>
</InvoiceDetailRequestHeader>
```

## 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.

## Example

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.

### Example

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."**

### 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>
  </TaxDetail>
</Tax>
```



## Invoices for suppliers from Mexico

When suppliers from Mexico (supplier country is Mexico) create invoices on Ariba Network, the `InvoiceDetailRequest` document must contain additional fields in the `TaxDetail`, `PostalAddress`, `InvoiceDetailRequestHeader` elements. The `TaxDetail` element also contains additional extrinsics.

### Note

Suppliers can integrate directly with an accredited electronic invoicing service provider to create CFDI invoices in the service provider's portal or generate CFDI invoices from their accounting system. Tax invoices for integrated Mexican suppliers use different extrinsics. See [Tax invoice extrinsics \[page 292\]](#).

The `TaxDetail` element contains the following element:

Element	Description
<code>TaxRegime</code>	<p>Specifies the tax category related to the type of supplier activities and commodities in invoices. Tax Regime is the governing organization for a tax category. Suppliers pay taxes based on the tax regime.</p> <p>Sample tax regimes:</p> <ul style="list-style-type: none"><li>• Assimilated Regime wages</li><li>• Regime of Small Taxpayers (REPECO)</li><li>• Regimen of business and professional activities, etc.</li></ul> <p>This is an optional element.</p> <p>An invoice can have one or more tax regimes, but suppliers can associate only one tax regime in the <code>TaxDetail</code> element. This element is applicable to all countries.</p>

```
<Tax>
  <Money currency = "USD">1.87</Money>
  <Description xml:lang = "en-US"/>
  <TaxDetail category="vat" percentageRate="2"
    taxPointDate="2013-06-19T00:00:00+05:30">
    <TaxableAmount>
      <Money currency = "USD">93.60</Money>
    </TaxableAmount>
    <TaxAmount>
      <Money currency = "USD">1.87</Money>
    </TaxAmount>
    <Description xml:lang = "en-US"/>
    <TaxRegime>Regimen de Asalariados</TaxRegime>
  </TaxDetail>
</Tax>
```

The `PostalAddress` element has the following element:

Element	Description
Municipality	<p>Specifies the name of the municipality for a division of the state in an Address' location. This is an optional element and added as part of the <code>PostalAddress</code> element.</p> <div><p><b>i Note</b></p><p>This field is mandatory only for suppliers with country Mexico and is used to digitally sign invoices from Mexico. If this information is missing in the <code>From</code> address element, the CFDI (legal invoice for Mexico) is incomplete and the service provider (EDICOM) and the Tax authority of Mexico does not approve the CFDI document.</p></div>

```
<PostalAddress>
  <Street>24 Mossy Creek</Street>
  <City>Chihuahua</City>
  <Municipality>Juárez</Municipality >
  <State>Chihuahua</State>
  <PostalCode>94089</PostalCode>
  <Country isoCountryCode = "MX">Mexico</Country>
</PostalAddress>
```

The `InvoiceDetailRequest` document also stores the `Contact role=billFrom`. This field is mandatory in invoices for suppliers with `From` and `To` country Mexico.

The following elements are also mandatory in the `PostalAddress` element:

- `Street`
- `City`
- `State`
- `PostalCode`

#### **i Note**

The `invoiceID` field can be used to store the folio number for the CFDI. The `invoiceID` field length must be restricted to 20 characters.

The `InvoiceDetailRequestHeader` has the following extrinsic:

Element	Description
<code>paymentMethod</code>	<p>Specifies the mode of payment. Possible values:</p> <ul style="list-style-type: none"> <li>• Cash</li> <li>• DirectDeposit</li> <li>• CreditTransfer</li> <li>• Wire</li> <li>• Check</li> <li>• Others</li> </ul> <p>This extrinsic information is stored in the <code>Contact</code> <code>role="remitTo"</code> in the <code>InvoiceDetailRequestHeader</code> element.</p> <p>Suppliers from Mexico must configure at least one <code>remitTo</code> address.</p> <pre>&lt;Extrinsic name =   "paymentMethod"&gt;creditTransfer&lt;/Extrinsic&gt;</pre>
<code>paymentNote</code>	<p>Specifies the number of installments that the buyer needs to pay the invoice. This value is taken from the purchase order, if it is available.</p> <p>Suppliers can enter a value for this extrinsic while creating an invoice. On Ariba Network, this field is available in the <b>Additional Fields</b> section on the invoice creation page.</p> <pre>&lt;Extrinsic name =   "paymentNote"&gt;1 installment   of 10&lt;/Extrinsic&gt;</pre>
<code>invoiceType</code>	<p>Specifies the invoice type. This extrinsic is added as part of the <code>InvoiceDetailRequestHeader</code> element.</p> <p>Possible values:</p> <ul style="list-style-type: none"> <li>• commercialInvoice</li> <li>• commissionInvoice</li> <li>• feelInvoice</li> <li>• rentInvoice</li> <li>• carrierInvoice</li> </ul> <p>This field is available in invoices for suppliers with <code>From</code> and <code>To</code> country Mexico.</p> <pre>&lt;Extrinsic name =   "invoiceType"&gt;commercialInvoice&lt;/Extrinsic&gt;</pre>

The `TaxDetail` element has the following extrinsic:

Element	Description
<code>WithholdingTaxType</code>	<p>Specifies if the tax category has the withholding tax. This extrinsic is added as part of the <code>TaxDetail</code> element.</p> <p>Possible values:</p> <ul style="list-style-type: none"> <li>ISR</li> <li>IVA</li> </ul> <p>This field is mandatory in invoices for suppliers with <code>From</code> and <code>To</code> country Mexico.</p>

```
<TaxDetail category="withholdingTax" percentageRate="-1">
  <TaxableAmount><Money currency="USD">99.95</Money></TaxableAmount>
  <TaxAmount>
    <Money currency="USD">-1.00</Money>
  </TaxAmount>
  <TaxLocation xml:lang="en-US">San Jose</TaxLocation>
  <Description xml:lang="en-US">Testing Purpose</Description>
  <Extrinsic name="withholdingTaxType">IVA</Extrinsic>
</TaxDetail>
```

The following are the possible errors that occur when invoices created by suppliers from Mexico (supplier country is Mexico) fail on Ariba Network:

Error	This error occurs when the.....
Payment method is missing	Extrinsic <code>paymentMethod</code> is missing.
Payment note is missing	Extrinsic <code>paymentNote</code> is missing.
Invoice Type is missing	Extrinsic <code>invoiceType</code> is missing.
Withholding tax type is missing	Extrinsic <code>withholdingTaxType</code> is missing with the <code>withholdingTax</code> tax category.
Tax Regime is missing	Element <code>TaxRegime</code> is missing in one of the <code>TaxDetails</code> element.
Municipality is missing in Supplier address	Element <code>Municipality</code> is missing in the <code>From</code> element.
Bill From Contact is missing	Invoice partner contact with the role " <code>BillFrom</code> " is missing in the invoice.
Invoice ID exceeds maximum length of 20 characters	Invoice ID of the invoice exceeds 20 characters. The ID must be less than or equal to 20.
A digital certificate is required	Digital certificate has not been uploaded by the supplier.
A valid digital certificate is required	Digital certificate that has been uploaded in the supplier's profile is invalid or has expired.
Buyer VAT id is missing	Buyer has not configured the VAT ID in their profile and the VAT ID is missing in the invoice.
The Supplier VAT ID is missing	Supplier has not configured the VAT ID in their profile and the VAT ID is missing in the invoice.
Street1 is missing in the Supplier address	Supplier has not provided the Street information in the Contact role="From" <code>PostalAddress</code> element.
State is missing in the Supplier address.	Supplier has not specified the state in the Contact role="From" <code>PostalAddress</code> element.

Error	This error occurs when the.....
Postal Code is missing in the Supplier address	Supplier has not specified the postal code in the Contact role="From" PostalAddress element.
City is missing in the Bill From address	Supplier has not specified the city in the Contact role="billFrom" PostalAddress element.

## Attachments with invoices

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. Buying organizations can configure Ariba Network to include attachments when forwarding invoices to Ariba Buyer 8.2 or later.

### Additional References

- cXML User's Guide at <http://www.cxml.org>

## 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.

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 **Send URLs to view attachments on Ariba Network** in their transaction configuration.

### Example

```
<Extrinsic name="AttachmentOnline">  
  https://service.ariba.com/ad/invoiceDetail?poID=1234&anp=Ariba  
</Extrinsic>
```

## Extrinsic elements

Suppliers can include various `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>
```

This example produces the following online text:

```
anyname:click me
```

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 SAP 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 SAP 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 Polish 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 Polish 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 Ariba Network user interface, 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*.

## Tax invoice extrinsics

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.

The following extrinsics are used to support additional tax invoice information required by the service provider when the invoice is legally signed. They are added in the `InvoiceDetailRequestHeader` element:

Extrinsic Name	Description
<code>economicActivityCode</code>	Numeric code assigned by the SSI related to the economic activity of the supplier (for Chile).
<code>economicActivityDescription</code>	Free-form text describing the supplier's economic activity.
<code>externalPurpose</code>	DTE document type (for Chile).
<code>externalVersion</code>	This element contains the tax invoice format. Possible values include NFE, DTE, CFDI, EDIXML, XML, PDF, or other.
<code>invoiceSubmissionMethod</code>	The method of submission to the Ariba Network, either Online, LegalInvoiceViaXML, or NFeViaXML (for Brazil).
<code>schemaVersion</code>	Dotted version number (for example, 3 . 2) of the XML invoice format for a country.
<code>signatureVerifier</code>	This element identifies the platform, which verifies the tax invoice signature and generates the signature verification report. Possible values include BM4, BNG and AribaNetwork. The element is not applicable to signed cXML invoices.
<code>signValidationDateTime</code>	Validation date and time.
<code>signerProvider</code>	The name of the signer of the tax invoice (for example, Self-Signed or {ProviderName}).
<code>signingCertificate</code>	X509 signing certificate.
<code>signingRequester</code>	The organization that requested that the invoice be signed (for example, AribaNetwork).



Extrinsic Name	Description
signingTime	Signing date and time.
taxInvoiceAttachmentName	Name for the file which contains the signature verified.
taxInvoiceFormat	Tax invoice format (for example, NFE, DTE, CFDI, or EDIXML).
taxInvoiceNumber	External tax invoice number.
taxInvoiceRepresentation	This element identifies whether the invoice is a tax invoice or not. If the value is Tax, the invoice is a tax invoice. This is used for recognizing external tax invoices.
taxInvoiceVerification	This element contains information about the signing status of the tax invoice. It has two possible values, Signed and Not Signed. This is used for processing external tax invoices.
taxStampingDate	Signing date and time.

The following cXML excerpt shows an example of extrinsic elements used for a Chilean tax invoice:

```
<Extrinsic name="buyerVatID">86160300-8</Extrinsic>
<Extrinsic name="supplierVatID">77546140-3</Extrinsic>
<Extrinsic name="invoiceSourceDocument">PurchaseOrder</Extrinsic>
<Extrinsic name="invoiceSubmissionMethod">Online</Extrinsic>
<Extrinsic name="resolutionNumber">89001</Extrinsic>
<Extrinsic name="resolutionDate">2014-10-22</Extrinsic>
<Extrinsic name="economicActivityCode">671929</Extrinsic>
<Extrinsic name="economicActivityDescription">EMPRESAS DE ASESORIA, CONSULTORIA
FINANCIERA Y DE APOYO AL GIRO</Extrinsic>
<Extrinsic name="taxInvoiceNumber">224880677</Extrinsic>
<Extrinsic name="taxInvoiceFormat">DTE</Extrinsic>
<Extrinsic name="externalSignerProvider">Signature</Extrinsic>
<Extrinsic name="schemaVersion">DTE-1.0</Extrinsic>
<Extrinsic name="taxInvoiceRepresentation">Tax</Extrinsic>
<Extrinsic name="taxInvoiceVerification">Signed</Extrinsic>
<Extrinsic name="externalPurpose">33</Extrinsic>
<Extrinsic name="signingCertificate">MIIDiz...</Extrinsic>
<Extrinsic name="taxStampingDate">2003-10-13T09:33:20</Extrinsic>
<Extrinsic name="signValidationDateTime">2015-01-09T08:03:36-08:00</Extrinsic>
<Extrinsic name="taxInvoiceAttachmentName">cid:SignedXML</Extrinsic>
<Extrinsic name="signingRequester">AribaNetwork</Extrinsic>
```

## 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.

## Extrinsics for invoices sent by suppliers from Brazil

When suppliers send an NFe or CTe invoice in an XML format to Ariba Network, some of the fields in the XML invoice are transformed to an extrinsic on Ariba Network.

The following are the extrinsics available for the NFe invoices:

Extrinsic	Description
NFeCertificateSerialNumber	The serial number issued by the signing authority of Brazil. This number is retrieved from the encoded base certificate in the NFe invoice.
buyer VATID	The CNPJ number of the Brazilian buyer.
supplier VATID	The CNPJ number of the Brazilian supplier.
paymentTermsCode	The details of the preferred mode of payment for the invoice. Following values indicate the preferred mode of payment: <ul style="list-style-type: none"> <li>0 denotes cash payments</li> <li>1 denotes payment in parts</li> <li>2 denotes others</li> </ul>
regioncode	The code of the state issuing the invoice.
transactiontype	The type of transaction chosen by the supplier for the invoice. For example, the transaction type includes operation tasks such as: transfer, return, import, assignment, and shipping.
customersPartNo	The code assigned to the product by the supplier.
invoiceDueDate	The expected date of invoice delivery to the buyer's system.
treasuryAdditionalInformation	Provides additional information of interest to the tax authority of Brazil.
SEFAZEnvironment	The details of the deployed SEFAZ environment. Following values indicate the specific SEFAZ environment: <ul style="list-style-type: none"> <li>1 denotes Production</li> <li>2 denotes Approval</li> </ul>

Extrinsic	Description
SEFAZSolutionAppVersion	The version number of the SEFAZ application.
externalInvoiceNumber	The invoice number generated from the supplier's electronic invoicing solution.
SEFAZDate	The date on which SEFAZ authorized the invoice.
protocolNumber	The number issued by the tax authority of Brazil to identify transactions such as authorization, denial, and NFe cancellations.
digitalSignatureValue	The digital signature code issued by the signing authority of Brazil.
SEFAZResponseCode	The authorization code received during SEFAZ validation.
SEFAZResponseDescription	The details of SEFAZ document processing status.
carrierNumberOfVolumes	The details of the total volume of goods dispatched in the carrier.
carrierSpecificationVolume	The detailed specification of the dispatched items for delivery.
carrierNetWeight	The total weight of the invoiced goods in the carrier.
carrierGrossWeight	The gross weight inclusive of the carrier and goods weight.
carrierTermsOfDelivery	The details of the terms of delivery for the goods in the carrier.
carrierInscriptionState	The details of the state registration code on the carrier.
carrierVehiclePlate	The details of the registration number affixed on the number plate of the vehicle.
carrierVehicleUFBoard	The code used to identify the details of the state where the vehicle is registered.
VAT ID	The CNPJ number of the carrier.
finalRecipient	The contact details of the final recipient of the goods.
transportAllowanceOrCharge	The travel expenses covered for shipping the goods.
invoiceReferences	<p>The details of the invoice.</p> <p>Contains the following domains in the IdReference element:</p> <ul style="list-style-type: none"> <li>• <code>internalControlNumber</code>: The 44 character access key number issued by the signing authority of Brazil to specify document series.</li> <li>• <code>originalInvoiceNumber</code>: The invoice number generated from the supplier's electronic invoicing solution or a third party service provider.</li> <li>• <code>NFeKeyFromHeader</code>: Same as the <code>internalControlNumber</code>.</li> <li>• <code>supplierSequenceNumber</code>: The invoice number generated from the supplier's electronic invoicing solution or a third party service provider.</li> <li>• <code>legalInvoiceSequence</code>: The fiscal document series.</li> </ul>

The following are the extrinsics available for the CTe invoices:

Extrinsic	Description
supplierVatID	The supplier's CNPJ number.

Extrinsic	Description
buyerVatID	The buyer's CNPJ number.
shipmentMethodOfPayment	The means of payment chosen for the shipped goods.
paymentTermsCode	The code associated with the type of payment selected by the supplier.
regionCode	The code of the state issuing the invoice.
transactionType	The type of transaction chosen by the supplier for the invoice.  For example, the transaction type includes operation tasks such as: transfer, return, import, assignment, and shipping.
invoiceReferences	The details of the invoice.  Contains the following domains in the <code>IdReference</code> element: <ul style="list-style-type: none"> <li>• <code>CTeType</code>: The details of the type of invoice.</li> <li>• <code>internalControlNumber</code>: The 44 character access key number issued by the signing authority of Brazil.</li> <li>• <code>originalInvoiceNumber</code>: The invoice number generated from the supplier's electronic invoicing solution or a third party service provider.</li> <li>• <code>CTeNumber</code>: Same as the <code>internalControlNumber</code>.</li> <li>• <code>legalInvoiceSequence</code>: The serial number of the fiscal document.</li> </ul>
operationFiscalCode	The code associated with the type of operation task selected for the invoice. For example, transport, delivery, import, and return.
netWeight	The details of the total weight of the shipped goods.
carrierVehiclePlate	The number displayed on the vehicle plate.
vehicleType	The type of vehicle used for shipment of goods.
transportTerms	The terms and conditions applied during the transport of invoiced goods.
route	The details of the route followed during the delivery of goods.
customersPartNo	The code assigned to the product by the supplier.
SEFAZEnvironment	The details of the deployed SEFAZ environment.  Following values indicate the specific SEFAZ environment: <ul style="list-style-type: none"> <li>• 1 denotes Production</li> <li>• 2 denotes Approval</li> </ul>
SEFAZSolutionAppVersion	The version number of SEFAZ application.
externalInvoiceNumber	The invoice number generated from the supplier's electronic invoicing solution.
SEFAZDate	The date on which invoice was authorized by SEFAZ.
protocolNumber	The number issued by the tax authority of Brazil to identify transactions such as authorization, denial, and NFe cancellations.
digitalSignatureValue	The digital signature code issued by the signing authority

Extrinsic	Description
SEFAZResponseCode	The authorization code received during SEFAZ validation.
SEFAZResponseDescription	The details of SEFAZ document processing status.
externalversion	The details of the version number of the CTe document.

The following extrinsics support tax invoicing for Brazilian service invoices:

Extrinsic	Description
rpsSeries	The alpha-numeric RPS series number.
rpsNumber	The RPS number.
verificationCode	The verification code provided by the municipality.
isNFSe	The information whether the document is an NFS-e invoice or not.
serviceCode	The five digit numeric service code to identify the service type..
taxationType	The taxation type eligible for the specific service invoice.
isISSRetention	This is an Yes/No field to indicate if the buyer is allowed to retain the ISS amount in the specific service invoice.
CNAECode	The code for economic activity.
regionalEngineerAssociationNum	The regional registration number issued to a supplier for performing construction related services.
culturalIncentive	This is a Yes/No field to indicate whether the supplier is eligible for cultural incentives.
simpleNational	This is a Yes/No field to indicate whether the supplier is a microentrepreneur or small business.
specialTaxRegime	The special regime identification code for taxation.
taxBenefitCode	The three digit code which indicates the eligibility for tax benefits.
issRetentionValue	The value indicates the amount retained for ISS tax.

The following excerpt from a cXML invoice displays the extrinsics that support tax invoicing for Brazilian service invoices:

```
<Extrinsic name="rpsSeries">BBBBB</Extrinsic>
<Extrinsic name="rpsNumber">110</Extrinsic>
<Extrinsic name="verificationCode">QACHQ</Extrinsic>
<Extrinsic name="isNFSe">yes</Extrinsic>
<Extrinsic name="serviceCode"><Classification domain="saoPaulo">123456</Classification></Extrinsic>
<Extrinsic name="taxationType">T</Extrinsic>
<Extrinsic name="isISSRetention">yes</Extrinsic><Extrinsic name="CNAECode">22222222</Extrinsic>
<Extrinsic name="regionalEngineerAssociationNum">89765478</Extrinsic>
<Extrinsic name="culturalIncentive">yes</Extrinsic>
<Extrinsic name="simpleNational">no</Extrinsic>
<Extrinsic name="specialTaxRegime">5</Extrinsic>
<Extrinsic name="taxBenefitCode">456</Extrinsic>
```

```
<Extrinsic name="issRetentionValue"><Money currency="USD">10.00</Money></Extrinsic>
```

## Extrinsics for invoices sent by Hungarian suppliers

### cXML changes

To support maintaining invoice number ranges for Hungarian suppliers, the following cXML extrinsic has been added to `InvoiceDetailRequestHeader`:

Extrinsic Name	This Field Stores...
taxInvoiceNumber	Numeric text that stores the current tax invoice number from the range set up by the supplier.

The following cXML excerpt shows the tax invoice number extrinsic in the header of an invoice.

```
<Request deploymentMode="production">
  <InvoiceDetailRequest>
    <InvoiceDetailRequestHeader invoiceDate="2016-02-03T16:41:04+08:00"
invoiceID="INV001" invoiceOrigin="supplier" operation="new" purpose="standard">
      ...
      <Extrinsic name="taxInvoiceNumber">101</Extrinsic>
      ...
    </InvoiceDetailRequestHeader>
  </InvoiceDetailRequest>
</Request>
```

## Extrinsic for invoices created automatically from receipts

The following cXML extrinsic contained in `InvoiceDetailRequestHeader` makes it possible to identify invoices that have been created automatically from receipts:

Extrinsic Name	This field stores...
isAutoflip	No value.

The following cXML excerpt shows the extrinsic element for invoices that have been created automatically from receipts:

```
<InvoiceDetailRequestHeader invoiceID="InvD012043"
purpose="standard" operation="new"
invoiceDate="2015-03-30T17:30:00-07:00">
  <InvoiceDetailHeaderIndicator></InvoiceDetailHeaderIndicator>
  ...
  <Extrinsic name="isAutoflip"/>
</InvoiceDetailRequestHeader>
```

## Extrinsics for additional information for payment terms

The Ariba Network provides several extrinsics related to payment terms information on invoices.

The `InvoiceDetailRequestHeader` element can contain the following cXML extrinsic elements to specify additional information for payment terms:

Extrinsic Name	Description
<code>discountInformation</code>	Unstructured information related to discounts.
<code>penaltyInformation</code>	Unstructured information related to penalties.
<code>netTermInformation</code>	Unstructured information related to net term.

The following cXML excerpt shows the extrinsic elements:

```
<Extrinsic name="discountInformation">No discounts are applied</Extrinsic>
<Extrinsic name="penaltyInformation">No penalties are applied</Extrinsic>
<Extrinsic name="netTermInformation">No net term is defined</Extrinsic>
```

## Extrinsic for referencing a payment proposal

Buyers can submit a `PaymentRemittanceRequest` to a supplier for invoices that do not exist on the procurement system. Each remittance detail of a specific payable that has been paid can reference a payment proposal using an extrinsic element named `PaymentProposalID`, as shown in the following cXML example.

```
<RemittanceDetail lineNumber="1">
  <PayableInfo>
    <PayableInvoiceInfo>
      <InvoiceIDInfo invoiceID="INV-skm-0504-1"
        invoiceDate="2011-05-04T04:22:23-07:00"/>
      <PayableOrderInfo>
        <OrderIDInfo orderID="PO-skm-0504-1"/>
      </PayableOrderInfo>
    </PayableInvoiceInfo>
  </PayableInfo>
  <NetAmount>
    <Money currency="USD">1039.90</Money>
  </NetAmount>
  <GrossAmount>
    <Money currency="USD">1039.90</Money>
  </GrossAmount>
  <DiscountAmount>
    <Money currency="USD">0</Money>
  </DiscountAmount>
  <AdjustmentAmount>
    <Money currency="USD">0</Money>
  </AdjustmentAmount>
  <Extrinsic name="PaymentProposalID">PPR-skm-0504-1</Extrinsic>
</RemittanceDetail>
```

## Extrinsics for country-specific fields in the invoice header

This topic shows how the country-specific fields from the buyer's and supplier's profiles map to cXML elements or extrinsic elements.

Field	Extrinsic	XPath
Supplier GST registration number	N/A	InvoiceDetailRequestHeader/ InvoicePartner/ Contact[@role='from']/IdReference domain gstID
Buyer GST registration number	N/A	InvoiceDetailRequestHeader/ InvoicePartner/ Contact[@role='soldTo']/IdReference domain gstID
Supplier PST registration number	N/A	InvoiceDetailRequestHeader/ InvoicePartner/ Contact[@role='from']/IdReference domain provincialTaxID
Buyer PST registration number	N/A	InvoiceDetailRequestHeader/ InvoicePartner/ Contact[@role='soldTo']/IdReference domain provincialTaxID
Supplier QST registration number	N/A	InvoiceDetailRequestHeader/ InvoicePartner/ Contact[@role='from']/IdReference domain qstId
Buyer QST registration number	N/A	InvoiceDetailRequestHeader/ InvoicePartner/ Contact[@role='soldTo']/IdReference domain qstId
Supplier legal form	LegalStatus	InvoiceDetailRequestHeader/ InvoicePartner/ Contact[@role='from']/ Extrinsic[@name='LegalStatus']
Supplier registered number	supplierCommercialIdentifier	InvoiceDetailRequestHeader/ Extrinsic[@name='supplierCommercial Identifier']



Field	Extrinsic	XPath
<b>Supplier commercial register court</b>	commercialRegisterCourt	InvoiceDetailRequestHeader/Extrinsic[@name='supplierCommercialRegisterCourt']
<b>Supplier late payment penalty</b>	penaltyInformation	InvoiceDetailRequestHeader/Extrinsic[@name='penaltyInformation']
<b>Supplier company capital</b>	LegalCapital	InvoiceDetailRequestHeader/Extrinsic[@name='LegalCapital']
<b>Supplier discount conditions</b>	discountInformation	InvoiceDetailRequestHeader/Extrinsic[@name='discountInformation']

## Invoice element field lengths

Invoice elements have maximum field lengths. Be aware of these maximums when creating invoices.

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
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)

Element or Attribute	Field Description
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 of invoices 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.

Ariba Network allows invoices to route only for buying organizations that have enabled their Ariba Network accounts for invoicing.

## Related Information

[Invoicing business rules \[page 247\]](#)

# Status and cancel invoices

Buying organizations can set invoice status. Suppliers can cancel invoices.

## In this section:

[Invoice status \[page 303\]](#)

[Cancel invoices \[page 304\]](#)

# Invoice status

Buying organizations send `StatusUpdateRequest` documents to set invoice status.

The following table lists valid invoice status settings and what they mean.

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.

SAP Ariba Buying and Invoicing downloads canceled invoices, evaluates them, and sends a `StatusUpdateRequest` document to Ariba Network. While SAP Ariba Buying and Invoicing evaluates canceled invoices, Ariba Network sets their document status to “canceling.” If SAP Ariba Buying and Invoicing approves them, Ariba Network sets their document status to “canceled.” If SAP Ariba Buying and Invoicing rejects them, Ariba Network sets their document status back to their original state.

## Example summary invoice

The summary invoice example illustrates a summary invoice against two different purchase orders.

The summary invoice 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`.

### **i** 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-supplierxyzkjlkwxxx-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>
            </InvoicePartner>
        </InvoiceDetailRequestHeader>
    </InvoiceDetailRequest>
</Request>

```

```

        </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>
<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="PO123">
            <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="PO123">
            <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>

```

```

        </NetAmount>
        <DueAmount><
            Money currency="USD">60.00</Money>
        </DueAmount>
    </InvoiceDetailSummary>
</InvoiceDetailRequest>
</Request>
</cXML>

```

## Example 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>
<Comments xml:lang="en-US">Sample Header Level Individual
    Invoice</Comments>
</InvoiceDetailRequestHeader>
<InvoiceDetailHeaderOrder>
    <InvoiceDetailOrderInfo>
        <OrderIDInfo orderID="DO21756"
            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>
                    </PostalAddress>
                </Contact>
            </InvoiceDetailShipping>
        </InvoiceDetailLineShipping>
    </InvoiceDetailOrderSummary>
</InvoiceDetailHeaderOrder>
</InvoiceDetailRequestHeader>

```

```

        <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>

```

## Example blanket purchase order invoice

The example blanket purchase order invoice illustrates 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>
      </InvoiceDetailRequestHeader>
    </InvoiceDetailRequest>
  </Request>
</cXML>
```

```

<InvoicePartner>
  <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>
    <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>

```

## Example of a complex buyer/supplier scenario

The example documents in the complex buyer/supplier scenario illustrate a buyer purchase order, a standard invoice, a credit memo, and a debit memo.

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).

## Example purchase order: complex buyer/supplier scenario

The example 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="DO12042"
        orderDate="2001-12-04T15:26:00-07:00" type="new"
        orderType="regular">
        <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>
          </Address>
        </ShipTo>
      </OrderRequestHeader>
    </OrderRequest>
  </Request>
</cXML>
```

```

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

```

## Example standard invoice: complex buyer/supplier scenario

The example standard 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="INVDO12042.120402" 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>
            <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>
        <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>
</Distribution>

```

```

        </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>
        <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>
            <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>

```

## Example credit memo: complex buyer/supplier scenario

The example credit memo illustrates how a supplier applies a credit including 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>
</InvoiceDetailRequest>
</Request>
</cXML>

```

## Example debit memo: complex buyer/supplier scenario

The example debit memo illustrates how a supplier corrects an error in a credit 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="DBDO12042-120402" 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="DBDO12042" purpose="debitMemo"
        operation="new" invoiceDate="2001-12-04T18:50:00-07:00">
        <InvoiceDetailHeaderIndicator isHeaderInvoice="yes"/>
        <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>
<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>

```

## Example cancel invoice

This example illustrates a cancel invoice. `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.

As always, this document has a unique `payloadID`.

```

<?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="PO123">
      <DocumentReference
        payloadID="1002700953000.152865612">
      </DocumentReference>
    </OrderReference>
  </InvoiceDetailOrderInfo>
  <InvoiceDetailItem invoiceLineNumber="1" quantity="10">
    <UnitOfMeasure>EA</UnitOfMeasure>
    <UnitPrice>
      .
      .
      .
    </UnitPrice>
  </InvoiceDetailItem>
</InvoiceDetailOrder>

```



---

## 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.

### 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>
```

## Dynamic discounting tax adjustments

Buyers can pass discount basis on payment proposals from their ERP to the Ariba Network, allowing SAP Ariba Discounting to exclude taxes and other potential ERP-defined elements from the discount calculation. Ariba Network passes discount amount, tax rebates, and revised settlement amounts back to the buyer and supplier ERPs when a supplier accepts a discount offer, and displays adjustments resulting from tax rebates clearly in the remittance advice. Buyers can configure the combinations of tax categories and tax location code (which can be mapped to ERP-defined country, region, or province) for which Ariba Network should calculate a tax rebate.

Discount basis must be equal to the taxable amount. You should not exclude other tax elements from the discount basis for countries in which tax adjustment applies.

Work with your IT department to configure your ERP to send tax adjustment information on payment proposals.

For additional information about configuring tax categories and tax location codes, see the *Payments Discounting Buyer's Guide*.

## Payment Term Offers

Payment Term Offers allow buyers and suppliers to agree to different payment terms at the invoice level when business requirements necessitate more flexibility on payment terms than standing payment terms allow. Ariba Network applies payment terms according to the `paymentTermCode` attribute that buyers include in cXML `PaymentProposalRequest` documents. If the `paymentTermCode` attribute matches a configured **Payment Term Offer** on Ariba Network, SAP Ariba Discounting applies that payment term. If the `paymentTermCode` attribute is not present or does not match a configured **Payment Term Offer** on Ariba Network, SAP Ariba Discounting instead uses any payment term that would have otherwise been applied. Example `PaymentProposalRequest` with `paymentTermCode` attribute:

```
<PaymentProposalRequest paymentDate="2016-08-05T11:59:20-07:00"
    paymentProposalID="5300000322" operation="new">
    <PayableInfo>
        <PayableInvoiceInfo>
            <InvoiceIDInfo invoiceID="INV322" invoiceDate="2016-07-06"/>
        </PayableInvoiceInfo>
    </PayableInfo>
</PaymentProposalRequest>
```

```

        </PayableInvoiceInfo>
    </PayableInfo>
    <PaymentMethod type="ach"/>
    <PaymentTerms paymentTermCode="210A"/>

```

Please work with your IT department to pass the `paymentTermCode` attribute in cXML `PaymentProposalRequest` documents. See *Payments and Discounting Buyer Guide* for more information about how to configure payment term offers.

## Example scheduled payment

The example scheduled payment demonstrates 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>

```

## Example scheduled payment with tax adjustments

The following is an example updated payment proposal that passes a discount basis and tax amount for the Goods and Services Tax (GST) of British Columbia:

```

<PaymentProposalRequest paymentProposalID="1100000124" operation="update"
paymentDate="2016-04-05T11:59:20-07:00">
    <PayableInfo>
        <PayableInvoiceInfo>
            <InvoiceIDInfo invoiceID="INV8291087"
invoiceDate="2016-04-01T18:31:35+05:30"></InvoiceIDInfo>
        </PayableInvoiceInfo>
    </PayableInfo>
    <PaymentMethod type="ach"></PaymentMethod>
    <GrossAmount>
        <Money currency="CAD">13440</Money>
    </GrossAmount>
    <DiscountBasis>
        <Money currency="CAD">12000</Money>
    </DiscountBasis>
    <DiscountAmount>
        <Money currency="CAD">1200</Money>
    </DiscountAmount>
    <AdjustmentAmount>
        <Money currency="CAD">0</Money>
    </AdjustmentAmount>
    <Tax>
        <Money currency="CAD">1440.00</Money>
        <Description xml:lang="en-US"></Description>
        <TaxDetail category="gst" percentageRate="5">
            <TaxableAmount>
                <Money currency="CAD">12000.00</Money>
            </TaxableAmount>
            <TaxAmount>
                <Money currency="CAD">600.00</Money>
            </TaxAmount>
        </TaxDetail>
    </Tax>

```

```

<Description xml:lang="en-US"></Description>
</TaxDetail>
<TaxDetail category="pst" percentageRate="7">
<TaxableAmount>
<Money currency="CAD">12000.00</Money>
</TaxableAmount>
<TaxAmount>
<Money currency="CAD">840.00</Money>
</TaxAmount>
<Description xml:lang="en-US"></Description>
</TaxDetail>
</Tax>
<TaxAdjustment>
<Money currency="CAD">60</Money>
<TaxAdjustmentDetail category="gst" location="BC">
<Money currency="CAD">60</Money>
</TaxAdjustmentDetail>
</TaxAdjustment>
<NetAmount>
<Money currency="CAD">12180</Money>
</NetAmount>
<Extrinsic name="Scheduling"></Extrinsic>
<Extrinsic name="Scheduled">yes</Extrinsic>
<Extrinsic name="immediatepay">yes</Extrinsic>
</PaymentProposalRequest>

```

---

# Receipts

Receipts indicate that your organization received ordered products or services. Your shipping clerks or requisitioners create them and send them to SAP Ariba Invoice Management through Ariba Network. Accounts payable personnel use them (in addition to purchase orders and invoices) to determine when and how much to pay suppliers.

## In this section:

[ReceiptRequest routing \[page 330\]](#)

[cXML ReceiptRequest document \[page 330\]](#)

[Example receipt document \[page 335\]](#)

## ReceiptRequest routing

Your organization generates receipts and addresses them to suppliers. Ariba Network routes them to SAP Ariba Invoice Management. 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 SAP Ariba Invoice Management and sends copies to the **Receipts** page in your online Outbox tab. It does not match receipts to corresponding documents. This allows you to send receipts for expired purchase orders or master agreements, or for purchase orders that were not routed through Ariba Network.

In SAP Ariba Invoice Management, users use the receipts (if required) and purchase orders to ensure the invoices they receive from suppliers are correct. See [Receiving types in SAP Ariba Invoice Management \[page 145\]](#) for information about 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.

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.

### Note

The `ReceiptRequest` document is defined in `Fulfill.dtd` rather than `cXML.dtd`.

## ReceiptRequestHeader

The `ReceiptRequestHeader` element contains header information for this receipt.

The `ReceiptRequestHeader` has the following attributes:

Attribute	Description
<code>receiptID</code> (required)	A buyer-generated unique identifier for this receipt.
<code>receiptDate</code> (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.
<code>operation</code>	The operation described by this receipt document. Must be "new".

## Comments

The optional `Comments` attribute enables you to provide comments with the receipt.

## Extrinsic

The optional `Extrinsic` element allows you to provide additional information related to a receipt. The information in an extrinsic element should not duplicate information elsewhere in the `ReceiptRequest` document.

## ReceiptOrder

The `ReceiptOrder` element defines information related to a purchase order or master agreement. A `ReceiptRequest` document may contain multiple `ReceiptOrder` elements.

The `ReceiptOrder` element has the following attribute:

Attribute	Description
<code>closeForReceiving</code>	<p>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.</p> <p>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.</p>

---

## 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:

Attribute	Description
<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.



Attribute	Description
quantity (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>
type	<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 `ReceiptItem` reference indicates the line number of a referenced line item.

`ReceiptItemReference` has the following attribute:

Attribute	Description
lineNumber (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  $\text{quantity} \times \text{UnitRate}$ .

## AssetInfo

The `AssetInfo` element contains optional asset data for each quantity of each receipt line item.

This element has the following attributes:

Attribute	Description
<code>tagNumber</code>	Internal tag number for this asset
<code>serialNumber</code>	Manufactures serial number for this asset
<code>location</code>	Location of this asset

## Comments

The `Comments` element enables you to add comments for this line item

## Total

The `Total` element provides a summary of the total amount of all receipt line item amounts.

## Money

The `Money` attribute specifies the currency type of the total money amount represented by the receipt.

## Example receipt document

The example receipt document 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="DO19">
                    <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>
        <Total>
            <Money currency="USD">301.00</Money>
        </Total>
    </ReceiptRequest>
</Request>
</cXML>

```

---

# Catalog upload transaction

The cXML Catalog Upload transaction enables suppliers to upload and publish catalogs on Ariba Network programmatically. The transaction provides an alternative to logging in to Ariba Network to upload and publish catalogs interactively. Use it to distribute updated catalogs automatically whenever you change pricing or availability of products or services.

Ariba Network supports any catalog in CIF, cXML, or BMEcat 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.

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.

## In this section:

[Catalog upload transaction overview \[page 338\]](#)

[CatalogUploadRequest prerequisites \[page 338\]](#)

[CatalogUploadRequest document \[page 340\]](#)

[Attaching your catalog \[page 343\]](#)

[Example of sending a CatalogUploadRequest document \[page 343\]](#)

[Receiving the response \[page 345\]](#)

[Receiving later catalog status \[page 346\]](#)

## Related Information

[Catalog upload transaction overview \[page 338\]](#)

[Example of sending a CatalogUploadRequest document \[page 343\]](#)

[Receiving the response \[page 345\]](#)

[Receiving later catalog status \[page 346\]](#)

# Catalog upload transaction overview

The Catalog Upload transaction consists of two cXML documents, `CatalogUploadRequest` and `Response`.

The purpose of these documents is as follows:

cXML Document	Purpose
<code>CatalogUploadRequest</code>	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.
<code>Response</code>	Sent by Ariba Network to acknowledge the receipt of a <code>CatalogUploadRequest</code> .

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.

## CatalogUploadRequest prerequisites

The `CatalogUploadRequest` transaction has requirements for your Ariba Network account, cXML-enabled applications, and cXML document authentication. It also requires valid catalogs.

### Ariba Network account

You must have a full-use supplier account on Ariba Network. If you have a light account, you'll need to upgrade to a full-use account before you can load the `CatalogUploadRequest` transaction. You can determine your account type by looking at the **Catalogs** tab in your supplier account - if it's accessible, you have a full-use account, but if it's grayed out, you have a light account. You can start the upgrade process by clicking **Learn More** while hovering over the **Catalogs** tab.

In addition to requiring a full-use account, you must be the account administrator 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-document authentication

You must have configured your Ariba Network account and your cXML-enabled application to [authenticate received cXML documents \[page 30\]](#).

## Valid catalogs

You must have valid catalogs. Catalogs can be in CIF 2.1, CIF 3.0, cXML, BMEcat, or Excel format.

To debug your catalogs as you develop them, use your Ariba Network account to upload them. Ariba Network automatically validates them and flags any syntactic errors. Your catalogs are validated when you use the `CatalogUploadRequest` transaction, but [solving catalog problems \[page 345\]](#) 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.

### Note

You can't use the `CatalogUploadRequest` transaction to publish new catalogs to buyers. You can publish a catalog using the `CatalogUploadRequest` transaction only if there is a previous version of the catalog already existing on Ariba Network.

However, in the case of BMEcat catalogs, you can use the `CatalogUploadRequest` transaction to publish both new catalogs as well as updates to existing catalogs to CMS-enabled buyers. These catalogs are not validated by Ariba Network and are automatically published to the CMS in the buyer's SAP Ariba solution. You do not have to log on to your account and manually publish these catalogs.

The following information is available in the **Learning Center**:

For information about...	See...
Creating CIF and cXML catalogs	<i>Catalog format reference</i>
Creating Excel format catalogs	<i>Creating and managing catalogs</i>
Uploading, validating, and testing catalogs	<i>Creating and managing catalogs</i>

---

# CatalogUploadRequest document

A `CatalogUploadRequest` document contains `To` and `CatalogUploadRequest`, and requires you to attach your valid catalog.

## 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. It also contains several elements that detail aspects of the request.

The `Operation` attribute is one of:

- `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.

## 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 <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:".

### Related Information

[Receiving the response \[page 345\]](#)

## 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.

### Related Information

[Recommended coding systems \[page 377\]](#)

## 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.

### Note

BMEcat catalogs are automatically published after upload irrespective of whether the `AutoPublish` element in the `CatalogUploadRequest` cXML document is set to `true` or `false`.

---

## Notification element

`Notification` sends catalog-status notifications through email or cXML POST.

For examples of these messages, see [Receiving later catalog status \[page 346\]](#).

`Notification` contains one or both of the `Email` element and `URLPost` 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.

`URLPost` has an `Enabled` attribute that specifies whether Ariba Network sends catalog-status notifications through `StatusUpdateRequest`:

- `true`: Enables this feature.
- `false`: Disables this feature.

## Related Information

[Profile transaction \[page 63\]](#)

---

## Attaching your catalog

Send your catalog attached to the `CatalogUploadRequest` document as MIME attachment. Large catalogs must be zipped to compress them before uploading.

## Use 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.

## Example of sending a `CatalogUploadRequest` document

The example of sending a `CatalogUploadRequest` document contains two documents enclosed in a MIME envelope: a `CatalogUploadRequest` document and a CIF 3.0 catalog.

```
--kdflkajfdksadjfklasdjfkljdfdsfdkf
Content-type: text/xml; charset=UTF-8
Content-ID: <part0.PCO28.975529413484@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--1234567899555556789@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>
--kdflkajfdksadjfklasdjfkljdfdsfdkf
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
--kdflkajfdksadjfklasdjfkljdfdsfdkf--

```

# Receiving the response

After you send a `CatalogUploadRequest`, Ariba Network replies with a standard cXML Response document.

## Example

```
<?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 on-line 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 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 notification of catalog status

If you specify an email mailbox to receive catalog status updates in `CatalogUpdateRequest`, then Ariba Network sends status notifications there.

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 notification of catalog status

If you specify URLPost to receive catalog status updates in `CatalogUpdateRequest`, then Ariba Network sends a cXML message to the URL you specified.

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>
  </Header>
</cXML>
```

```

    </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:

Status Code	Meaning
200 Success	The catalog-upload request succeeded.
463 Bad Catalog Format	The zip file is invalid.
470 Catalog Has Errors	The message is the status of the catalog. (HasErrors)

# Supply chain financing trades

TradeRequest cXML documents are used for the supply chain financing discount method. Each cXML TradeRequest corresponds to a Trade document on the Ariba Network.

Trade documents are associated with PaymentProposalRequest documents that are traded on the PrimeRevenue trading platform. payment proposals can be traded automatically or manually as indicated by the autoTrade attribute of the TradeRequestHeader.

TradeRequest documents travel from the supply chain financing provider to the Ariba Network. Updated PaymentProposalRequests are then generated by the Ariba Network and sent to the buyer and supplier as needed.

A TradeRequest has the following basic structure:

- TradeRequestHeader: indicates the buyer, supplier, and third-party funder
- TradeRequestSummary: indicates the original value of the scheduled payment, credits and fees, and the amount traded.
- TradeItems that indicate the specific payables that have been traded to third-party funders.

The following is an example of a TradeRequest document:

```
<TradeRequest>
  <TradeRequestHeader operation="new" status="accepted" tradeID="trade200922"
    tradeDate="2015-09-30T10:43:36-07:00"
    tradeApprovedDate="2015-09-30T12:43:36-07:00"
    settlementDate="2015-10-02T10:43:36-07:00" autoTrade="no">
    <PaymentPartner>
      <!--Funder information-->
      <Contact role="payer">
        <Name xml:lang="en">Bank Of America</Name>
        <IdReference domain="financialInstitutionID" identifier="987654321">
        </IdReference>
      </Contact>
    </PaymentPartner>
    <PaymentPartner>
      <!-- supplier information -->
      <Contact role="payee">
        <Name xml:lang="en">Supplier Name</Name>
      </Contact>
    </PaymentPartner>
    <Contact>
      <Name xml:lang = "en">John Smith</Name>
      <Phone> <!--optional -->
        <TelephoneNumber>
          <CountryCode isoCountryCode = "BE">32</CountryCode>
          <AreaOrCityCode/>
          <Number>0477 07 26 41</Number>
        </TelephoneNumber>
      </Phone>
    </Contact>
    <Comments>Optional comments</Comments>
  </TradeRequestHeader>
  <TradeRequestSummary>
    <!-- trade level totals -->
    <OriginalAmount> <!-- sum of all OriginalAmount of all TradeItems -->
      <Money currency="USD">1000</Money>
    </OriginalAmount>
```



```

    <CreditApplied>      <!-- sum of all credit applied of all TradeItems -->
      <Money currency="USD">100</Money>
    </CreditApplied>
    <FeeAmount>          <!-- total fee -->
      <Money currency="USD">20</Money>
    </FeeAmount>
    <Amount>             <!-- total projected amount to be paid to supplier -->
      <Money currency="USD">880</Money>
    </Amount>
  </TradeRequestSummary>
  <TradeItem lineNumber="1" paymentProposalID="PPR910"
    maturityDate="2015-04-30T10:43:36-07:00">
    <PayableInfo>
      <PayableInvoiceInfo>
        <InvoiceIDInfo invoiceDate="2015-03-30T10:43:36-07:00"
          invoiceID="330inv1"></InvoiceIDInfo>
        </PayableInvoiceInfo>
      </PayableInfo>
      <OriginalAmount> <!-- payment amount of the original PPR -->
        <Money currency="USD">1000</Money>
      </OriginalAmount>
      <AdjustmentAmount>
        <Money currency="USD">100.00</Money>
      <Modifications>
        <Modification>
          <AdditionalDeduction type="creditApplied">
            <DeductionAmount>
              <Money currency="USD">100.00</Money>
            </DeductionAmount>
          </AdditionalDeduction>
        </Modification>
      </Modifications>
    </AdjustmentAmount>
    <DaysPaidEarly>10</DaysPaidEarly>
    <Amount> <!-- trade amount: GrossAmount - AdjustmentAmount - FeeAmount-->
      <Money currency="USD">900</Money>
    </Amount>
    <!-- fee information, optional and not in credit memo -->
    <FeeAmount>
      <Money currency="USD">15.00</Money> <!-- total fee -->
      <Fee type="serviceProvider">
        <Money currency="USD">5.00</Money>
      </Fee>
      <Fee type="community">
        <Money currency="USD">5.00</Money>
      </Fee>
      <Fee type="funder">
        <Money currency="USD">5.00</Money>
      </Fee>
    </FeeAmount>
  </TradeItem>
  <TradeItem lineNumber="2" paymentProposalID="PPR911">
    maturityDate="2015-04-30T10:43:36-07:00"> <!-- credit memo -->
    <PayableInfo>
      <PayableInvoiceInfo>
        <InvoiceIDInfo invoiceDate="2015-03-30T10:43:36-07:00"
          invoiceID="cm123"></InvoiceIDInfo>
        </PayableInvoiceInfo>
      </PayableInfo>
      <OriginalAmount>
        <Money currency="USD">-100</Money>
      </OriginalAmount>
      <AdjustmentAmount>
        <Money currency="USD">-100.00</Money>
      <Modifications>
        <Modification>
          <AdditionalDeduction type="creditApplied">
            <DeductionAmount>

```

```
        <Money currency="USD">-100.00</Money>
      </DeductionAmount>
    </AdditionalDeduction>
  </Modification>
</Modifications>
</AdjustmentAmount>
<DaysPaidEarly>0</DaysPaidEarly>
<Amount>
  <Money currency="USD">0</Money>
</Amount>
</TradeItem>
</TradeRequest>
```

---

# Get pending/data download transaction

Ariba Network buyers and suppliers can receive documents from Ariba Network by polling for them and downloading them.

Most Ariba Network buyers and suppliers use cXML routing to push incoming documents over the Internet to their cXML application on the web. Optionally, buyers and suppliers can also use polling and downloading to receive data from Ariba Network. In the polling scenario, the Ariba Network queues data by document type for downloading initiated by applications later. Ariba Buyer, non-Ariba procurement applications, and 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.

Users 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.

## **i** Note

In SAP Ariba Buying and Invoicing, Ariba Network sends the document to SAP Ariba Buying and Invoicing by posting the document. By posting the document, customers can immediately view order confirmations, ship notices, invoices, and other documents sent by suppliers. SAP Ariba Buying and Invoicing does not use the cXML get pending and download transactions.

### **In this section:**

- [Polling frequency \[page 352\]](#)
- [Polling time \[page 352\]](#)
- [Download all waiting documents \[page 352\]](#)
- [Confirm the receipt of documents \[page 353\]](#)
- [GetPendingRequest \[page 353\]](#)
- [GetPendingResponse \[page 355\]](#)
- [DataRequest \[page 356\]](#)
- [DataResponse \[page 357\]](#)

## 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.

Procurement or supplier application can download these documents:

- `StatusUpdateRequest`
- `ConfirmationRequest`
- `ShipNoticeRequest`
- `ServiceEntryRequest`
- `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* in the help 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 [page 358] 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`.

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. Generally, you should 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 the application times out due to excessive attachment sizes, limit the `maxMessages` attribute to a number less than 10, for example:

```

<GetPendingRequest lastReceivedTimestamp="2015-12-01T16:00:00-08:00"
  maxMessages="1">
  <MessageType>InvoiceArchive</MessageType>
</GetPendingRequest>

```

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>
  </Header>
</cXML>

```

```

    </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>
          <Sender>
            <Credential domain="NetworkID">
              <Identity>AN01000000001</Identity>
            </Credential>
            <UserAgent>ANCXMLDispatcher</UserAgent>
          </Sender>
        </Header>
        <Message>
          <DataAvailableMessage>
            <InternalID domain="PendingMessages">3738</InternalID>
          </DataAvailableMessage>

```

```

        </Message>
      </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.

### i 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 [page 63].

```

<?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 included together in a MIME envelope using 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--
```

## About 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. Therefore 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 using the `lastReceivedTimeStamp` attribute:

1. The procurement application sends a `GetPendingRequest`:

```
<GetPendingRequest lastReceivedTimeStamp="time1"/>
```

Ariba Network responds with

```
<cXML timestamp="time2">
  <GetPendingResponse>
    <DataAvailableMessage>
      <InternalID domain="PendingMessages">id1</InternalID>
    </DataAvailableMessage>
  </GetPendingResponse>
</cXML>
```

2. The procurement application retrieves the waiting documents by sending a `DataRequest`:

```
<DataRequest>
  <InternalID domain="PendingMessages">id1</InternalID>
</DataRequest>
```

Ariba Network responds with

```
<DataResponse>...</DataResponse>
```

At this point, the procurement application can download `id1` 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 `id1` and the procurement application can no longer download `id1`.

---

Ariba Network responds with

```
<cXML timestamp="time3">
  <GetPendingResponse>
    <DataAvailableMessage>
      <InternalID domain="PendingMessages">iid2</InternalID>
    </DataAvailableMessage>
  </GetPendingResponse>
</cXML>
```

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 TLS server certificate when they initiate HTTPS connections.

# About SAP Ariba Supply Chain Collaboration

SAP Ariba Supply Chain Collaboration is an extension of Ariba Network that enhances the ability of buyers to collaborate with suppliers in supply-chain tasks such as ordering, invoicing, and shipping.

With SAP Ariba Supply Chain Collaboration enabled, an array of features become available to buyers and suppliers on Ariba Network. These features facilitate collaboration between buyers and suppliers, for example on transactions pertinent to direct material supply chains.

As a buyer, the SAP Ariba Supply Chain Collaboration solution allows you to do the following:

- Collaborate efficiently with all supply chain partners through a single global solution.
- Improve visibility into supply with continuous monitoring, benchmarking, and enhancement of supplier performance
- Easily manage all direct materials procurement and typical retail and manufacturing collaborative processes - including consigned inventory, rolling delivery schedules, and contract manufacturing - all in one place
- Respond intelligently to changes in supply and demand with real-time visibility into supply chain metrics
- Rapidly connect to thousands of suppliers with minimal IT costs
- Instantly identify errors through configurable business rules with automated validation and reconciliation

## Note

SAP Ariba Supply Chain Collaboration features entail enhancements to the Ariba Network user interface. Fields and tabs that do not appear in the regular Ariba Network interface may appear in the SAP Ariba Supply Chain Collaboration user interface. Also, the same tab may be labeled with one name in the regular Ariba Network user interface, and with another in the SAP Ariba Supply Chain Collaboration user interface. Users should take note of these differences, especially when they first switch from the regular Ariba Network user interface to the SAP Ariba Supply Chain Collaboration user interface.

### In this section:

[Consignment collaboration \[page 360\]](#)

[Business scenarios and assumptions for Forecast and Commit B2B \[page 363\]](#)

## Consignment collaboration

A consignment sale means that goods are sent from the supplier to the buyer, but the supplier retains ownership of the consignments until the buyer pays for the consignments the buyer consumes through a pre-arranged purchase-order method. Consignments stocked in the buyer's warehouses or production facilities are not paid for until they have been used. The transfer of ownership, and often of location, to the buyer at the time of consumption is called a **consignment movement**.

SAP Ariba Supply Chain Collaboration provides ways for suppliers to view a list of materials on consignment, for suppliers to create invoices for consignment movements, and for buyers to create self-billing invoices for consignment movements.

The following elements support consignment inventory collaboration:

Element	Contained In	Description
ProductActivityMessage	n/a	Transmits component inventory, consignment movement, and forecast information from the buyer's ERP system. The buyer-provided inventory summary view includes the components issued to the supplier. The information provided gives a snapshot of the component inventory and forecast situation at a certain point in time.
ProductActivityHeader	ProductActivityMessage	Contains information about a single component inventory, the product forecast details, or a consignment movement for the product.
ProductActivityDetails	ProductActivityMessage	Contains the details about a product activity.
Inventory	ProductActivityDetails	Inventory that is in the possession of the buyer, and is owned and managed by the buyer.
ConsignmentInventory	ProductActivityDetails	Inventory that is in the possession of the buyer, but is owned by the supplier.
ConsignmentMovement	ProductActivityDetails	Contains information about the movement of material from the consignment inventory to the buyer inventory.
ProductMovementItemIDInfo	ConsignmentMovement	A reference to the line item in a movement document.
InvoiceItemIDInfo	ConsignmentMovement	Line item of an invoice created by the buyer against the movement item.

## ConsignmentMovement example

The following example shows a ProductActivityMessage used to communicate the movement of material from the consignment inventory to the buyer inventory:

```
<ProductActivityMessage>
  <ProductActivityHeader messageID="PRO ACT_1001"
creationDate="2014-11-10T22:00:00-08:00"/>
  <ProductActivityDetails>
    <ItemID>
      <SupplierPartID>ZEEEE25</SupplierPartID>
      <BuyerPartID>REEEA25</BuyerPartID>
    </ItemID>
    <Description xml:lang="en">Movement of Consignment</Description>
    <Batch productionDate="2014-11-10T22:00:00-08:00"
expirationDate="2014-12-10T22:00:00-08:00" originCountryCode="US">
      <BuyerBatchID>BWWBRCA25</BuyerBatchID>
      <SupplierBatchID>SWWBRCA25</SupplierBatchID>
    <PropertyValuation>
    <PropertyReference>
      <IdReference identifier="CHEMICAL" domain="ID"/>
      <IdReference identifier="1" domain="VersionID"/>
    </PropertyReference>
  </ProductActivityDetails>
</ProductActivityMessage>
```

```

        <IdReference identifier="MEDICINE_PROPERTIES"
domain="PropertyDefinitionClassReferenceID"/>
        <IdReference identifier="1"
domain="PropertyDefinitionClassReferenceVersionID"/>
        </PropertyReference>
        <ValueGroup>
            <IdReference identifier="2" domain="ID"/>
        </ValueGroup>
    </PropertyValuation>
</Batch>
<Contact role="locationTo">
    <Name xml:lang="en">Palo Alto - Plant1</Name>
    <IdReference domain="buyerLocationID" identifier="0001">
    <Description xml:lang="en">Sunnyvale Plant</Description >
    </IdReference>
</Contact>
<ConsignmentMovement>
    <ProductMovementItemIDInfo movementLineNumber="3"
movementID="MADOC421304_r" movementDate="2015-03-10T11:00:00-08:00"/>
    <MovementQuantity quantity="200">
        <UnitOfMeasure>TOK</UnitOfMeasure>
    </MovementQuantity>
    <SubtotalAmount>
        <Money currency="USD">2400</Money>
    </SubtotalAmount>
    <UnitPrice>
        <Money currency="USD">12</Money>
    </UnitPrice>
</ConsignmentMovement>
<ConsignmentMovement>
    <ProductMovementItemIDInfo movementLineNumber="4"
movementID="MADOC421305_r" movementDate="2015-03-12T11:00:00-08:00"/>
    <MovementQuantity quantity="500">
        <UnitOfMeasure>TOK</UnitOfMeasure>
    </MovementQuantity>
    <SubtotalAmount>
        <Money currency="USD">6000</Money>
    </SubtotalAmount>
    <UnitPrice>
        <Money currency="USD">12</Money>
    </UnitPrice>
</ConsignmentMovement>
<ConsignmentMovement>
    <ProductMovementItemIDInfo movementLineNumber="5"
movementID="MADOC421306_r" movementDate="2015-03-14T11:00:00-08:00"/>
    <InvoiceItemIDInfo invoiceLineNumber="1" invoiceID="INVD1000" /
>
    <MovementQuantity quantity="300">
        <UnitOfMeasure>TOK</UnitOfMeasure>
    </MovementQuantity>
    <SubtotalAmount>
        <Money currency="USD">3600</Money>
    </SubtotalAmount>
    <UnitPrice>
        <Money currency="USD">12</Money>
    </UnitPrice>
</ConsignmentMovement>
</ProductActivityDetails>
</ProductActivityMessage>

```

---

# Business scenarios and assumptions for Forecast and Commit B2B

Following are the business scenarios and assumptions for Forecast and Commit.

## Forecast

The forecast message is always considered a full snapshot of a product. The occurrence of the same combination of product and plant in the cxml will overwrite earlier data for that product and plant.

Any consecutive forecasts received will also be considered as overwrites if the same product exists in the system.

The forecast quantity will be overwritten in the database if the product already exists in the database. This is determined by the following factors:

- If the buyer part id, supplier part id and plant match with an existing record.
- If the buyer part id and plant match with an existing record, and the supplier part id is different it will be overwritten.
- If the supplier part id and plant match with an existing record where the buyer part id is null and the current product has the buyer part id, then the buyer part id will be updated to the same record.

A new record will be created in the database if the product does not exist in the system. This is determined by the following factors:

- If the buyer part id does not match with the existing record
- If the plant does not match with the existing record

The forecast data will be overwritten from the current date or the earliest date that has come in the cxml. This means that the past data will be retained and all the future quantities will be zeroed.

If the same date appears for a product within the product details of the product, then the quantities will be aggregated.

Java calendar is considered for all the date calculations.

For weekly aggregation, a week is considered as running Monday to Sunday.

Java calendar will consider the last few days of the year to be part of first week of next year in cases of years having more than 52 weeks.

For daily calculation we consider only the date and ignore the timestamp and the timezone.

The overwrite/update of forecast is done by moving the existing forecast quantity to the change column and moving the current quantity in the forecast quantity column.

If a forecast document comes with an empty start date, then all the forecast data for that product, past and present, will be removed. However, the product will still be present in the system.

---

## Commit

The commit message is handled similar to the forecast document.

The commit message also is considered a full snapshot for a product.

A consecutive commit message will overwrite the commit and upside quantities for the product.

The update is done based on the following. Check to see if the product exists in the system:

- If the buyer part id, supplier part id and plant match with an existing record.
- If the buyer part id and plant match with an existing record and the supplier part id is different, it will be overwritten

In no case is a new record created by a commit message. This means the commit quantities will be saved in the system only if there is an existing forecast for that product.

Rules for commit calendar and date are similar to those for forecast calendar and date.

If a commit document comes with an empty start date, then the commit data will be cleared from the current date or the earliest date from the cxml. It will not clear the forecast data.



---

# Contracts

If SAP Ariba Contracts is integrated with an SAP ERP, users can send items from the line items document and project header fields in a procurement contract workspace to the ERP.

When a user selects the **Send to External System** option from a line items document menu, SAP Ariba Contracts sends the line items and information from the project header fields to the ERP. The information is sent in a cXML `ContractRequest` document to Ariba Network, and Ariba Network forwards the request to the ERP. The ERP sends back status information in a cXML `ContractStatusUpdateRequest` document to Ariba Network, and Ariba Network forwards the request to SAP Ariba Contracts.

## In this section:

[ContractRequest](#) [page 365]

[ContractStatusUpdateRequest](#) [page 369]

## ContractRequest

SAP Ariba sends a `ContractRequest` document to an SAP ERP when a user selects **Send to External System** from the menu of a contract line items document. The **Send to External System** menu option is available when the line items document and the workspace are both published.

A `ContractRequest` document contains information from the contract workspace header and the contents of the line items document, including the `type` element, which specifies the type of SAP contract document to create (value `contract` or `quantity contract`). The value of the `type` element is determined by the **Document Category** field for the contract workspace.

The operation in a `ContractRequest` document is `new` the first time the line items document is sent to the ERP. If the ERP successfully processes the `ContractRequest`, the ERP sends back a `ContractStatusUpdateRequest` document with the `Status Succeeded`, and `IDReference` elements with the ID of the SAP contract document and corresponding line numbers.

If the line items document has already been sent to the SAP ERP and the user re-sends the document, the operation is `update`. SAP Ariba sends `ReferenceDocumentInfo` with values from the `IDReference` elements in the `ContractStatusUpdateRequest` to identify the SAP contract document and line items.

## SAP service items

A `ContractRequest` document can contain data that SAP uses to create SAP service items (SAP Item Category D items) that have a single-level hierarchy.

SAP Ariba sends a parent `ContractItemIn` and child `ContractItemIn` pair for each contract line item with the term `Item Category` set to `D` (services). A child `ContractItemIn` element is linked to a parent

ContractItemIn element using the parentLineNumber attribute in the ItemIn element; the ContractItemIn element with the matching lineNumber attribute in its ItemIn element is the parent.

The ItemIn element in a ContractItemIn element for a service line item also contains an ItemClassification attribute set to service.

The ItemDetail element for a service line item includes a Classification element; the Classification element's domain attribute and value are mapped to an SAP Ariba Strategic Sourcing solutions commodity code domain and value.

## Example

```
<Request deploymentMode="production">
  <ContractRequest>
    <ContractRequestHeader
      operation="new"
      xml:lang="en"
      expirationDate="2016-01-30T00:00:00-00:00"
      effectiveDate="2016-01-11T00:00:00-00:00"
      type="value"
      agreementDate="2016-01-12T00:00:00-00:00"
      createDate="2016-01-11T23:36:18+08:00"
      contractID="CW2009">
      <LegalEntity domain="CompanyCode">100</LegalEntity>
      <OrganizationID>
        <Credential domain="NetworkID">
          <Identity>AN02000000120</Identity>
        </Credential>
        <Credential domain="sap">
          <Identity>0000000100</Identity>
        </Credential>
      </OrganizationID>
      <OrganizationalUnit domain="PurchasingOrganization">
        1001
      </OrganizationalUnit>
      <OrganizationalUnit domain="PurchasingGroup">
        10101
      </OrganizationalUnit>
      <PaymentTerm payInNumberOfDays="10">
        <Discount>
          <DiscountPercent percent="2"></DiscountPercent>
        </Discount>
        <Extrinsic name="Id">0001</Extrinsic>
      </PaymentTerm>
      <MaxAmount>
        <Money currency="USD">2000.00</Money>
      </MaxAmount>
      <TermsOfDelivery>
        <TermsOfDeliveryCode value="TransportCondition"/>
        <ShippingPaymentMethod value="Other"/>
        <TransportTerms value="FOB">Free on board vessel</TransportTerms>
      </TermsOfDelivery>
    </ContractRequestHeader>
    <ContractItemIn>
      <TermsOfDelivery>
        <TermsOfDeliveryCode value="TransportCondition"/>
        <ShippingPaymentMethod value="Other"/>
        <TransportTerms value="FOB">Free on board vessel</TransportTerms>
      </TermsOfDelivery>
      <ItemIn lineNumber="1" quantity="100" itemClassification="material">
      <ItemID>
```

```

    <SupplierPartID>1</SupplierPartID>
    <SupplierPartAuxiliaryID></SupplierPartAuxiliaryID>
    <BuyerPartID>992</BuyerPartID> <!-- Material code -->
  </ItemID>
  <ItemDetail>
    <UnitPrice>
      <Money currency="USD">1000.00</Money>
      <Modifications>
        <Modification>
          <AdditionalDeduction type="DISCOUNT">
            <DeductionAmount>
              <Money currency="USD">10.00</Money>
            </DeductionAmount>
          </AdditionalDeduction>
        </Modification>
        <Modification>
          <AdditionalDeduction type="DISCOUNT">
            <DeductionPercent percent="20"/>
          </AdditionalDeduction>
        </Modification>
        <Modification>
          <AdditionalCost>
            <Money currency="USD">30.00</Money>
          </AdditionalCost>
        </Modification>
        <Modification>
          <AdditionalCost>
            <Percentage percent="20"/>
          </AdditionalCost>
        </Modification>
      </Modifications>
    </UnitPrice>
    <Description xml:lang="en">Laptops</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="unspsc">43211503</Classification>
    <Classification domain="MaterialGroup">29</Classification>
    <ManufacturerPartID></ManufacturerPartID>
    <ManufacturerName></ManufacturerName>
    <URL></URL>
    <LeadTime>2</LeadTime>
  </ItemDetail>
  <ShipTo>
    <Address>
      addressID="3000"
      addressIDDomain="buyerLocationID"
      isoCountryCode="US">
      <Name xml:lang="en" >Plant 3000</Name>
    </Address>
  </ShipTo>
</ItemIn>
<ReferenceDocumentInfo lineNumber = "10">
  <DocumentInfo documentID = "PR1234"
    documentType = "Requisition"
    documentDate = "2015-11-07T07:03:34-05:00">
  </DocumentInfo>
</ReferenceDocumentInfo>
<ReferenceDocumentInfo lineNumber = "3">
  <DocumentInfo documentID = "RFQ2345"
    documentType = "RFQ"
    documentDate = "2015-11-07T07:03:34-05:00">
  </DocumentInfo>
</ReferenceDocumentInfo>
</ContractItemIn>
</ContractRequest>
</Request>

```

## ContractRequestHeader

ContractRequestHeader is the header element for ContractRequest. It has the following attributes:

Attribute	Description
contractID (required)	SAP Ariba Contracts contract ID for this request.
type	Identifies whether the contract is value-based or quantity-based. Possible values are "value" or "quantity".
createDate	The date and time the contract workspace was created.
agreementDate	<b>Agreement Date</b> for the workspace.
effectiveDate (required)	<b>Effective Date</b> for the workspace.
expirationDate	<b>Expiration Date</b> for the workspace.
xml:lang (required)	The language or locale in which the ContractRequest content is written.
operation	The operational mode of the ContractRequest. Possible values are: <ul style="list-style-type: none"><li>• new—Identifies a new contract transaction.</li><li>• update—Identifies an update to an existing transaction. The DocumentInfo element is used to indicate the contract in the external system.</li></ul>

ContractRequestHeader has the following elements:

Element	Description
LegalEntity (required)	A legal entity in the ERP. It has an IdReference element.
OrganizationID (required)	Provides credentials for the organization ID.
OrganizationalUnit	Identifies the Purchase Unit or Purchase group in the ERP. It has an IdReference element.
PaymentTerm	Defines a payment term in an invoice or order. Payment term can be the net term (without discount) or discount term (with discount).
QuoteRequestReference	Reference to a quote request originated in the ERP.
MaxAmount	The maximum amount for the contract.
MinAmount	The minimum amount for the contract.
MaxReleaseAmount	The contractual maximum quantity per release of a contract.
MinReleaseAmount	The contractual minimum quantity per release of a contract.
Contact	Supplies additional Address or Location information for the requesting company.
Comments	Any additional comments about the contract request.
DocumentInfo	Contract ID managed in the ERP. Included if the operation is "update".
ParentContractInfo	Parent contract ID from the ERP if the current contract is part of a hierarchy.

Element	Description
TermsOfDelivery	Optional shipping terms (incoTerms) as defined by the International Chamber of Commerce.
Extrinsic	Additional information about the contract request header.

## ContractItemIn

ContractItemIn represents a contract line item to be sent to the ERP. It has the following elements:

Element	Description
MaxAmount	The maximum amount for an item.
MinAmount	The minimum amount for an item.
MaxReleaseAmount	The contractual maximum quantity for an item per release (order).
MinReleaseAmount	The contractual minimum quantity for an item per release (order).
MaxQuantity	The maximum quantity for an item.
MinQuantity	The minimum quantity for an item.
MaxReleaseQuantity	The contractual maximum quantity for an item per release (order).
MinReleaseQuantity	The contractual minimum quantity for an item per release (order).
TermsOfDelivery	Optional shipping terms (incoTerms) as defined by the International Chamber of Commerce.
ItemIn (required)	An item from the source buyer system.
ReferenceDocumentInfo	Optional reference document info for this line item. For example, the Requisition or RFQ in the ERP.

## ContractStatusUpdateRequest

The SAP ERP sends a ContractStatusUpdateRequest document after it receives a ContractRequest from SAP Ariba.

The ContractStatusUpdateRequest document contains the Status element to indicate if the ERP successfully created or updated a corresponding contract document. It also contains IDReference elements inside ContractIDInfo and ContractItemStatus elements to identify the corresponding document and items on the ERP.

## Example

```
<Request deploymentMode="production">
  <ContractStatusUpdateRequest>
    <Status xml:lang="en-US" code="200" text="OK">Succeeded</Status>
    <ContractStatus type="created">
      <ContractIDInfo contractID="CW2009">
        <IdReference identifier="55000000" domain="SAPAgreementId"/>
      </ContractIDInfo>
      <ContractItemStatus>
        <ItemStatus type="created">
          <ReferenceDocumentInfo lineNumber="1"/>
        </ItemStatus>
        <IdReference identifier="010" domain="SAPLineNumber"/>
      </ContractItemStatus>
      <ContractItemStatus>
        <ItemStatus type="created">
          <ReferenceDocumentInfo lineNumber="2"/>
        </ItemStatus>
        <IdReference identifier="020" domain="SAPLineNumber"/>
      </ContractItemStatus>
    </ContractStatus>
  </ContractStatusUpdateRequest>
</Request>
```

## Status

Status of a Response or Message. It has the following attributes:

Attribute	Description
code (required)	HTTP or cXML-specific status code.
text (required)	Textual version of the status code.
xml:lang	The language or locale in which the ContractStatusUpdateRequest content is written..

## ContractStatus

ContractStatus contains item-level status updates for a contract. It has the following attribute:

Attribute	Description
type (required)	Type of the contract status, for example, "created".

---

ContractStatus has the following elements:

Element	Description
ContractIDInfo (required)	The contract ID information created/updated in the source buyer system.
ContractItemStatus	Represents a line item in a contract status update request.
Comments	Optional field for communicating arbitrary comments or description of an item.

---

# cXML 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.

## In this section:

[Turning on transformations \[page 372\]](#)

[Moving extrinsics to the Contact element \[page 372\]](#)

[Transforming ItemOut extrinsics \[page 374\]](#)

[Consolidating ItemOut to OrderRequestHeader \[page 375\]](#)

[Mapping custom extrinsics to Ariba Network extrinsics \[page 375\]](#)

[Mapping extrinsics from cXML to EDI \[page 376\]](#)

## 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.

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 extrinsics 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>
  </PostalAddress>
</Contact>
```

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 extrinsics

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 this extrinsic 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>
  . . .
</ItemOut>
```

Ariba Network moves this extrinsic 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 SAP 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 extrinsics from cXML to EDI

If you enable **Map standard Ariba Network extrinsics 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 extrinsics in purchase orders, or map custom extrinsics to standard Ariba Network extrinsics in purchase orders, Ariba Network converts those extrinsics to equivalent X12 REF elements. See the Ariba Network documentation for the list of standard Ariba Network extrinsics.

By default, this control is off, so Ariba Network maps extrinsic data to EDI comment elements in purchase orders.

# Recommended coding systems

SAP Ariba applications make use of standard coding systems for commodity, currency, unit of measure, country, language, and telephone dialing codes.

## In this section:

[Commodity codes \[page 377\]](#)

[Currency codes \[page 378\]](#)

[Unit of measure codes \[page 378\]](#)

[Country codes \[page 379\]](#)

[Language codes \[page 379\]](#)

[Dialing codes \[page 380\]](#)

## Commodity codes

The United Nations Standard Products and Services Code (UNSPSC) standard is the SAP Ariba-preferred commodity coding system. The UNSPSC is free and contains descriptions of more than 12,000 products and services.

The UNSPSC coding structure is hierarchical. It combines similar items in 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:  
<https://www.eccma.org/unspsc>
- United Nations Development Programme (UNDP). For a list of codes, go to:  
<http://www.unspsc.org>

UNSPSC 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

Code	Meaning
82111502	Manual writing services

## Currency codes

Specify currency names with ISO 4217 three-letter (CodeA) currency codes.

For a list of codes ISO 4217 currency codes, go to:

<http://www.currency-iso.org/en/home/tables/table-a1.html>

Click **Table A.1 (XLS)** to download a Microsoft Excel file with the currency codes.

### 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:

[http://www.unece.org/cefact/codesfortrade/codes\\_index.html](http://www.unece.org/cefact/codesfortrade/codes_index.html)

## 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:

<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:

<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:

### 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:

[https://www.itu.int/itudoc/itu-t/ob-lists/icc/e164\\_763.html](https://www.itu.int/itudoc/itu-t/ob-lists/icc/e164_763.html)

### Examples

Code	Meaning
55	Brazil
81	Japan
1	United States



# Revision history

The following table provides a brief history of the updates to this guide. SAP Ariba updates the technical documentation for its cloud solutions if

- 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.

SAP 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.

Month/year of update	Updated chapter/section	Short description of change
November 2014	All	Updated structure and format.
December 2014	Invoices and Scheduled Payments	Updated the Numeric Validation section to move it to its own section and add that there is no validation that line-level <code>SubtotalAmounts</code> add up to the header-level <code>SubtotalAmount</code> on incoming cXML invoices.
April 2015	Document Addressing and Security	Replaced SSL protocol with TLS.
July 2015	Purchase Orders	Added or updated the following topics: <ul style="list-style-type: none"><li>• Modifications Element</li><li>• Subcontracting Orders</li><li>• External Line Numbers</li><li>• Service Order with Multi-Level Hierarchy</li></ul>
	Order Confirmations and Ship Notices	Added the following topic: Confirmations and Service Orders.
	Service Sheets	Added the following topic: Automatically-Generated Service Sheets.

Month/year of update	Updated chapter/section	Short description of change
	Invoices and Scheduled Payments	<p>Added or updated the following topics:</p> <ul style="list-style-type: none"> <li>• PaymentTerm</li> <li>• compositeltemType Attribute</li> <li>• Allowances and Charges</li> <li>• level Attribute</li> <li>• InvoiceHeaderModifications</li> <li>• TotalAmountWithoutTax</li> <li>• Money or Percentage Allowances and Charges</li> <li>• ModificationDetail Names</li> <li>• French Parafiscal Taxes</li> <li>• Invoices for Suppliers From Mexico</li> <li>• Mandatory Remit To</li> <li>• Mandatory Supplier Information</li> <li>• Tax Invoice Extrinsics</li> <li>• Extrinsics for Invoices Sent by Suppliers from Brazil</li> <li>• Extrinsic for Invoices Created Automatically from Receipts</li> <li>• Extrinsics for Additional Information for Payment Terms</li> </ul>
	Consignment Collaboration	New chapter
August 2015	Invoices and Scheduled Payments	Updated “ACH Information” and “Wire Information” examples.
	Multiple chapters	Fixed typos.
October 2015	Requests for Quotation	Added new topic: “SAP Ariba Sourcing Integration to SAP ERP.”
	Purchase Orders	Added the three new extrinsics related service lines
November 2015	Invoices and Scheduled Payments	Corrected example in “Credit Memos and Line-Item Credit Memos.”
	Purchase Orders	Updated “Extrinsic for Legacy Purchase Orders.”
	Order Confirmations and Ship Notices	Fixed formatting of example in “Example OrderStatusRequest Document.”
	Invoices and Scheduled Payments	Added new topic, “Extrinsic for Referencing a Payment Proposal.”
December 2015	Get Pending/Data Download Transaction	Added information about <code>maxMessages</code> to “GetPendingRequest” topic.
	Order Confirmations and Ship Notices	Updated “Suppliers Account Configuration.”
	PunchOut Site Planning	In the “PunchOut Message Flow” topic, changed “Transfer Basket to Ariba Buyer” to “Checkout”.
	Invoices and Scheduled Payments	Updated “Maximum Document Size” and “AttachmentOnline Extrinsic” topics.
January 2016	Introduction to cXML Solutions	Revised “About cXML Versions” topic and removed “cXML Versions Used by Ariba Applications” topic.

Month/year of update	Updated chapter/section	Short description of change
	Supply Chain Collaboration	Added topic, Business Scenarios and Assumptions for Forecast and Commit B2B.
	n/a	Updated multiple sections for AN 15s release.
February 2016	Purchase Orders	Updated "Extrinsics for Service Periods" topic.
March 2016	Purchase Orders	<ul style="list-style-type: none"> <li>Updated "Masking Values in Blanket Purchase Orders and Service Purchase Orders" topic.</li> <li>Added "Incoterms on Orders" topic.</li> <li>Added "AribaNetwork.PaymentTermsExplanation Extrinsic" topic.</li> <li>Updated "Tax Element" and "Modifications Element" topics with information about taxes, charges and discounts sent by SAP Ariba Buying solutions.</li> </ul>
	Invoices and Scheduled Payments	<ul style="list-style-type: none"> <li>Updated "Tax Detail Category," "Tax on Shipping and Special Handling," and "Allowances and Charges" topics.</li> </ul>
	Purchase Orders	Added information about Funds Management accounting fields.
	Requests for Quotation	<ul style="list-style-type: none"> <li>Divided content into two sub-sections: "Sourcing Integration with ERP" and "Sourcing Integration with SAP Ariba Buying solutions."</li> <li>Added relevant information in the "Sourcing Integration with SAP Ariba Buying solutions" section.</li> <li>Added "Canceling Requests for Quotation" topic.</li> </ul>
	Invoices and Scheduled Payments	Added new topic, "Extrinsics for Invoices Sent by Hungarian Suppliers" under "Extrinsic Elements" section.
April 2016	Introduction to cXML Solutions	Moved "Maximum Document Size" topic to "Introduction to cXML Solutions" chapter.
	Purchase Orders	
	Invoices and Scheduled Payments	
May 2016	Contracts	New chapter.
	Receipts	Updated DTD filename in "cXML ReceiptRequest Document" topic.
	Tolerances Element	Updated for lower and upper tolerances.
	Purchase Orders	Updated "Planned Service Line Items" example.
	Get Pending/Data Download Transaction	Revised chapter topic.

Month/year of update	Updated chapter/section	Short description of change
	Invoices and Scheduled Payments	Added the extrinsics for invoices sent by suppliers from Brazil.
June 2016	cXML Data Conventions	Added new topic, "timestamp Attribute," under "Date and Time Format" topic.
August 2016	Requests for Quotation	Revised the chapter topic for quote automation related content. Added a topic on processingTarget extrinsic.
	Document Addressing and Security	Updated the the references to TLSv1 to reflect the support for TLSv1.1 and TLSv1.2.
	Invoices and Scheduled Payments	Added information about Discount Basis and Tax Category and Location.
November 2016	Supply Chain Financing Trades	New chapter.
January 2017	<ul style="list-style-type: none"> <li>Requests for Quotation</li> <li>Contracts</li> </ul>	Added information for service items.
February 2017	Invoices and scheduled payments	Added "Extrinsics for country-specific fields in the invoice header."
	Multiple chapters	Updated titles to use sentence capitalization.
March 2017	PunchOut transactions	Updated "Example PunchOutSetupResponse document" topic.
May 2017	Order confirmations and ship notices	Added following topics: <ul style="list-style-type: none"> <li>ShipNoticeHeader extrinsics</li> <li>Ship notice packaging dimensions</li> <li>Canceling a ship notice using cXML</li> </ul>
	Quick enablement	Updated "AttachmentOnline extrinsic" topic.
	Invoice and Scheduled Payments	Added information about the paymentTermCode attribute for payment term offer discounts.
	Catalog upload transaction	Added information about BMEcat catalogs.
June 2017	Multiple chapters	Updated SAP Ariba solution names.
August 2017	Purchase orders	Updated "AttachmentOnline extrinsic" topic, adding note about replacing escaped & characters in the URL.
November 2017	SAP Ariba Sourcing integration with SAP ERP	Added information about service items in service hierarchies.

---

# Important Disclaimers and Legal Information

## Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

## Accessibility

The information contained in the SAP documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of willful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

## Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

## Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: <https://help.sap.com/viewer/disclaimer>).



**www.ariba.com**

© 2017 SAP SE or an SAP affiliate company. All rights reserved.  
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.  
Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.  
These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.  
SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.  
Please see <https://www.sap.com/corporate/en/legal/copyright.html> for additional trademark information and notices.