



Litle & Co.

XML Reference Guide

March 2012

XML Release: 8.10

Document Version: 2.18a

The People Behind Your Payments

Litle & Co. XML Reference Guide Document Version: 2.18a

All information, whether text or graphics, contained in this manual is proprietary information of Litle & Co. and is provided to you solely for the purpose of assisting you in using a Litle & Co. product. All such information is protected by copyright laws and international treaties. The possession, viewing, or use of the information contained in this manual does not transfer any intellectual property rights or grant a license to use this information or any software application referred to herein for any purpose other than that for which it was provided. Information in this manual is presented "as is" and neither Litle & Co. or any other party assumes responsibility for typographical errors, technical errors, or other inaccuracies contained in this document. This manual is subject to change without notice and does not represent a commitment on the part of Litle & Co. or any other party. Litle & Co. does not warrant that the information contained herein is accurate or complete.

All trademarks are the property of their respective owners and all parties herein have consented to their trademarks appearing in this manual. Any use by you of the trademarks included herein must have express written permission of the respective owner.

Copyright © 2003-2012, Litle & Co. ALL RIGHTS RESERVED.

CONTENTS

About This Guide

Intended Audience	xv
Revision History	xv
Document Structure	xviii
Documentation Set	xviii
Typographical Conventions	xix
Contact Information.....	xx

Chapter 1 Introduction

The LittleXML Data Format.....	2
Communications Protocols	3
General XML Coding Requirements	3
Other XML Resources.....	4
Batch Transaction Processing	5
Duplicate Transaction Detection	6
Batch Duplicate Checking	6
Online Duplicate Checking.....	7
Coding for Report Groups.....	9
Smart Authorization Features	11
Prepaid Indicator	11
Affluence Indicator	12
Issuer Country Indicator.....	12
Cardholder Type Indicator.....	12
Card Filtering Services.....	13
Prepaid Card Filtering Service	13
International Card Filtering Service	14
Chargeback Filtering	14
Authorization Recycling	15
Recycling Engine	15
Recycling Advice.....	16
Authorization Recycling Best Practices.....	16
Tokenization Feature	17
How Tokenization Works	17
Little Token Formats	19
Obtaining Tokens.....	19
Bulk Token Registration	20
Supported Token Transactions	20
Compliance with Visa Best Practices for Tokenization	21

eCheck Processing	23
Validation Feature	23
Verification Feature	23
Required Contents of Decline Notice	23
Automatic Notice of Change (NoC) Updates	25
Auto Redeposit Feature	25
Automatic Account Updater Service	26
Automatic Account Updater Service Options	27
Full Service Option	27
Match and Repair Option	29
Merchant Requirements	30
Automatic Account Updater Features	30
Healthcare Card Feature	31
Supported Transaction Types	34
Authorization Transaction	34
AVS Only Transaction	35
Authorization Reversal Transactions	35
Notes on the Use of Authorization Reversal Transactions	36
Using Authorization Reversal to Halt Recycling Engine	37
Capture Transaction	38
Capture Given Auth Transaction	38
Credit Transaction	39
eCheck Credit Transaction	39
eCheck Redeposit Transaction	39
eCheck Sales Transaction	40
eCheck Verification Transaction	40
eCheck Void Transaction (Online Only)	40
Force Capture Transaction	40
Sale Transaction	40
Void Transaction (Online Only)	41
Using Void to Halt Recycling Engine	41

Chapter 2 Testing Your LittleXML Transactions

Certification Environment Usage Policy	44
Limitations	44
Regularly Scheduled Maintenance Window	44
Data Retention Policy	45
Overview of Testing	46
Planning for Certification Testing	46
Required Certification Testing	47
Optional Testing	47

Transferring Files	48
Transferring Batch Files	48
Submitting a Batch File for Processing	48
Retrieving Processed Batch Files	49
Transferring Online Files	49
ASP Programming Example	50
Java Programming Example	51
Notes on Timeout Settings	52
Helpful Web Sites	52
Performing the Required Certification Tests	53
Testing Authorization, AVS Only, Capture, Credit, Sale, and Void Transactions	53
Testing Recycling Advice	68
Testing Authorization Reversal Transactions	69
Testing eCheck Transactions	73
Testing Token Transactions	79
Performing the Optional Tests	85
Testing AVS and Card Validation	85
Testing Address Responses	87
Testing Advanced AVS Response Codes	89
Testing Response Reason Codes and Messages	90
Testing 3DS Responses	92
Testing the Prepaid Filtering Feature	94
Testing the International Card Filter Feature	96
Testing Automatic Account Updater	97
Testing Automatic Account Updater Extended Response Codes	98
Testing Automatic Account Updater for Tokenized Merchants	99
Testing Tax Billing and Convenience Fee	100
Testing the Recycling Engine	100
Testing Recycling Engine Cancellation	103
Testing Transaction Volume Capacity	103

Chapter 3 LittleXML Transaction Examples

Overview of Online and Batch Processing Formats	106
Batch Process Format	106
Supported Communication Protocols	106
Batch Processing Request Format	107
Batch Processing Response Format	109
Online Processing Format	110
Supported Communication Protocols	110
Online Processing Request Format	111
Online Processing Response Format	112

Transaction Types and Examples.....	113
Authorization Transactions.....	114
Authorization Request Structure	114
Authorization Response Structure	120
Authorization Reversal Transactions	124
Authorization Reversal Requests	124
Authorization Reversal Responses	125
Capture Transactions.....	127
Capture Request	127
Capture Response	134
Capture Given Auth Transactions	135
Capture Given Auth Request	135
Capture Given Auth Response	140
Credit Transactions	142
Credit Request for a Little & Co. Processed Transaction.....	142
Credit Request for a Non-Little & Co. Processed Transaction	146
Credit Response	150
eCheck Credit Transactions.....	152
eCheck Credit Request Against a Little & Co. Transaction.....	152
eCheck Credit Request for a Non-Little & Co. Processed Sale	154
eCheck Credit Response	155
eCheck Redeposit Transactions	156
eCheck Redeposit Request	156
eCheck Redeposit Response.....	157
eCheck Sale Transactions	159
eCheck Sale Request	159
eCheck Sale Response.....	160
eCheck Verification Transactions.....	162
eCheck Verification Request.....	162
eCheck Verification Response	164
eCheck Void Transactions (Online Only)	166
eCheck Void Request	166
eCheck Void Response.....	166
Force Capture Transactions.....	167
Force Capture Request.....	167
Force Capture Response	171
Register Token Transactions	173
Register Token Request	173
Register Token Response.....	175
RFR Transactions (Batch Only)	177
RFR Request	177

RFR Response.....	178
Sale Transactions	179
Sale Request.....	179
Sale Response.....	183
Void Transactions (Online Only)	186
Void Request.....	187
Void Response.....	187

Chapter 4 LittleXML Elements

accNum.....	190
accountInformation	191
accountNumber.....	192
accountUpdate.....	193
accountUpdateFileRequestData	194
accountUpdater.....	195
accountUpdateResponse.....	199
accType	200
actionReason	201
addressIndicator	202
addressLine1, addressLine2, addressLine3	203
advancedAVSResults	204
affiliate.....	205
affluence	206
allowPartialAuth	207
amexAggregatorData	208
amount	209
approvedAmount.....	210
authAmount.....	211
authCode	212
authDate	213
authenticatedByMerchant	214
authentication.....	215
authenticationResult	216
authenticationTransactionId	217
authenticationValue	218
authInformation	219
authorization	220
authorizationResponse	221
authorizationSourcePlatform.....	222
authReversal.....	223
authReversalResponse.....	224

availableBalance	225
avsResult	226
batchRequest	227
batchResponse	231
billMeLaterRequest	232
billMeLaterResponseData	234
billToAddress	235
bin	237
bmlMerchantId	238
bmlProductType	239
bypassVelocityCheck	240
campaign	241
capability	242
capture	243
captureGivenAuth	244
captureGivenAuthResponse	245
captureResponse	246
card	247
cardAcceptorTaxId	248
cardholderAuthentication	249
cardholderId	250
cardOrToken	251
cardProductType	252
cardValidationNum	253
cardValidationResult	254
chargeback	255
checkNum	256
city	257
clinicOtherAmount	258
code	259
commodityCode	260
companyName	261
country	262
credit	263
creditLine	265
creditResponse	266
customBilling	267
customerInfo	269
customerIpAddress	270
customerReference	271
customerRegistrationDate	272

customerType	273
customerWorkTelephone	274
deliveryType	275
dentalAmount	276
descriptor	277
destinationCountryCode	278
destinationPostalCode	279
detailTax	280
discountAmount	281
dob	282
dutyAmount	283
echeck	284
eCheckAccountSuffix	285
echeckCredit	286
echeckCreditResponse	287
echeckForToken	288
echeckOrEcheckToken	289
echeckRedeposit	290
echeckRedepositResponse	291
echeckSale	292
echeckSalesResponse	293
echeckToken	294
echeckVerification	295
echeckVerificationResponse	296
echeckVoid	297
echeckVoidResponse	298
email	299
employerName	300
enhancedAuthResponse	301
enhancedData	303
entryMode	305
expDate	306
extendedCardResponse	307
filtering	308
firstName	309
forceCapture	310
forceCaptureResponse	311
fraudResult	312
fundingSource	313
healthcareAmounts	314
healthcareIIAS	315

IIASFlag	316
incomeAmount	317
incomeCurrency	318
international	319
invoiceReferenceNumber	320
issuerCountry	321
itemCategoryCode	322
itemDescription	323
itemDiscountAmount	324
itemSequenceNumber	325
lastName	326
lineItemData	327
lineItemTotal	329
lineItemTotalWithTax	330
litleOnlineRequest	331
litleOnlineResponse	332
litleRequest	333
litleResponse	334
litleSessionId	335
litleToken	336
litleTxnId	337
merchantData	338
merchantGroupingId	339
merchantId	340
message	341
middleInitial	342
name	343
newAccountInfo	344
newCardInfo	345
newCardTokenInfo	346
newTokenInfo	347
nextRecycleTime	348
number	349
orderDate	350
orderId	351
orderSource	352
originalAccountInfo	353
originalCard	354
originalCardInfo	355
originalCardTokenInfo	356
originalToken	357

originalTokenInfo	358
password	359
payerId	360
paypage	361
paypageRegistrationId	362
paypal	363
paypalOrderComplete	364
phone	365
phone as a child of billToAddress and shipToAddress	365
phone as a child of customBilling	365
pos	366
postDate	367
postDay	368
preapprovalNumber	369
prepaid	370
prepaidCardType	371
processingInstructions	372
productCode	373
quantity	374
recycleAdvice	375
recycleAdviceEnd	376
recycleBy	377
recycling	378
recyclingRequest	379
registerTokenRequest	380
registerTokenResponse	381
reloadable	382
residenceStatus	383
response	384
responseTime	385
RFRRequest	386
RFRResponse	387
routingNum	388
RxAmount	389
sale	390
saleResponse	392
salesTax	393
sellerId	394
sellerMerchantCategoryCode	395
shipFromPostalCode	396
shippingAmount	397

shipToAddress	398
ssn	399
state	400
taxAmount	401
taxExempt	402
taxIncludedInTotal	403
taxRate	404
taxType	405
taxTypeIdentifier	406
termsAndConditions	407
token	408
token (Little generated card number replacement)	408
token (PayPal generated)	408
tokenMessage	409
tokenResponse	410
tokenResponseCode	411
totalHealthcareAmount	412
track	413
transactionId	414
type	415
type Element as a child of the parent elements listed below	415
type Element as a child of fundingSource	416
unitCost	417
unitOfMeasure	418
updatedCard	419
updatedToken	420
url	421
user	422
verificationCode	423
verify	424
visionAmount	425
virtualAuthenticationKeyData	426
virtualAuthenticationKeyPresenceIndicator	427
void	428
voidResponse	429
yearsAtEmployer	430
yearsAtResidence	431
zip	432

Appendix A Payment Transaction Response Codes

Payment Transaction Response Codes	434
--	-----

Visa Authentication Result Codes.....	447
AVS Response Codes	449
AAVS Response Codes.....	450
Card Validation Response Codes.....	453
XML Validation Error Messages	454
eCheck Return Reason Codes	456
eCheck NoC Change Codes.....	460

Appendix B Credit Card Number Formats



ABOUT THIS GUIDE

This manual serves as a reference to the LitleXML transaction formats used for payment processing with Litle & Co. It also explains how to perform unattended transaction testing and attended certification testing with Litle & Co.

Intended Audience

This document is intended for technical personnel who will be setting up and maintaining payment processing using the LitleXML format.

Revision History

This document has been revised as follows:

TABLE 1 Document Revision History

Doc. Version	Description	Location(s)
1.0	Initial release of combined (Batch + Online) and restructured XML Reference Guide.	N/A
1.1	Added information about the AMEX <advancedAVSResults> element and AAVS response Codes. Also, updated all examples to reflect change to LitleXML version 7.3. Also, added info to Note for <paypal> element in chapter 4.	Chapters 3 and 4. Appendix A.
1.2	Added a note specifying that orphaned Credits are not supported for PayPal transactions. Also, removed <payerEmail> from Chapter 4, since it is only used for PayPal orphaned Credits.	Chapters 3 and 4.
2.0	Added new information about echeck functionality.	All
2.1	Added new information about the echeck Void and eCheck redeposit transactions, including required test cases for certification testing.	All

TABLE 1 Document Revision History

Doc. Version	Description	Location(s)
2.2	Corrected information concerning fields required when requesting an eCheck Verification.	Chapters 2, 3, 4 and Appendix A
2.3	Removed “Batch Only” note for various eCheck transactions. Clarified information in Online Dupe Checking section. Fixed several errors in Test/Cert scenarios and added test cases for Enhanced Authorization features.	Chapters 1, 3 and 4 Chapter 1 Chapter 2
2.4	Added information and tests for eCheck Tokenization.	All
2.5	Added information about new elements used for Healthcare Card processing. Added Register Token examples to Chapter 3. Added 2 new AAVS Response Codes	All Chapter 3 Appendix A
2.6	Removed information about Amex Auth Reversals. Added “Certification Environment Usage Policy” section to Chapter 2. Removed namespace (xmlns) attribute from all Response examples. This hasn’t been used since schema version 7.0.	Chapter 1 and Appendix A Chapter 2 Chapters 3 and 4
2.7	Removed Response Code 803 Clarified language about the operation of Online duplicate transaction checking for referenced transactions.	Chapter 4 and Appendix A Chapter 1
2.8	Removed <orderId> element from Online Capture Response example. Added info to element definitions. Added information about new Smart Authorization elements. Added Information about Filtering Services. Changed Account Number on eCheck test cases that should have returned a Return Reason Code of 301 - Invalid Account Number.	Chapters 3 and 4 All All Chapter 2
2.9	Add information for new <cardProductType> element. Add information for new elements associated with Pay Page. Added descriptions to the Reason Response Codes table. Added info on using <customerIpAddress> element for AAVS.	All All Appendix A Chapter 4

TABLE 1 Document Revision History

Doc. Version	Description	Location(s)
2.10	Misc. minor corrections and clarifications. Added information about Duplicate Transaction Detection. Added Amex AuthReversal test case Added paypageRegistrationId option to Register Token Transactions and new Response Code (879).	All Chapters 1 and 4 Chapter 2 Chapter 3 and Appendix A
2.11	Added information about Automatic Account Updater service. Added information about new elements for recycling advice. Add new PayPal Response Codes (613 thru 628) Removed redepositCount element.	All Chapters 1, 3, and 4 Appendix A Chapters 3 and 4
2.12	Fixed typo in Testing chapter; ".acs" should have been ".asc" Fixed error in Figure 1-4. With Full option, new card data is not returned to the merchant.	Chapter 2 Chapter 1
2.13	Added Response Code 629; removed code 353	Appendix A
2.14	Restructured/rewrote Chapter 2. Added new codes and info for AMEX Advanced AVS. Added the paypage element as an option to the Capture Given Auth, Credit, and Force Capture transactions.	Chapter 2 Appendix A Chapters 3 and 4
2.15	Added Prepaid Indicator (5) and AAVS Cert tests. Added information about Recycling Engine. Added new schema elements for V8.8 Added new Response Codes	Chapter 2 All Chapters 2, 3, & 4 Appendix A
2.16	Added Recycling Engine Cert tests Added new schema element (<actionReason>) for V8.9 maxDigits for <amount> element changed from 8 to 12	Chapter 2 Chapters 3 and 4 Chapter 4
2.17	Added new schema elements for Auth Recycling control Added new response Reason Codes	Chapters 3 and 4 Appendix A
2.18	Removed notes about <chargeback> element not being used. Added section about Chargeback Filtering. Added Appendix B providing info about account number formats.	Chapter 4 Chapter 1 Appendix B
2.18a	New version created for posting on Little API web page.	footers and copyright page

Document Structure

This manual contains the following sections:

Chapter 1, "Introduction"

This chapter provides an introduction to transaction processing using LittleXML.

Chapter 2, "Testing Your LittleXML Transactions"

This chapter provides information concerning the testing and certification process, which you must complete prior to submitting transactions to the Little production environment.

Chapter 3, "LittleXML Transaction Examples"

This chapter provides information concerning the LittleXML structure for transaction submission, as well as examples.

Chapter 4, "LittleXML Elements"

This chapter provides definitions and other information concerning each LittleXML element.

Appendix A, "Payment Transaction Response Codes"

This appendix lists all of the possible response codes and messages.

Appendix B, "Credit Card Number Formats"

This appendix provides information about credit card number formats and Mod-10 validation.

Documentation Set

For additional information concerning the Little & Co. application, see any of the following guides in the documentation set:

- *Little & Co. User Interface Guide*
- *Little & Co. Chargeback XML and Support Documentation API Reference Guide*
- *Little & Co. Chargeback Process Guide*
- *Little & Co. PayPal Integration Guide*
- *Little & Co. Bill Me Later Integration Guide*
- *Little & Co. Virtual Terminal User Guide*
- *Little & Co. Pay Page Integration Guide*
- *Little & Co. XML Differences Guide*

Typographical Conventions

Table 2 describes the conventions used in this guide.

TABLE 2 Typographical Conventions

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
< >	Angle brackets are used in the following situations: <ul style="list-style-type: none">• user-supplied values (variables)• XML elements
[]	Brackets enclose optional clauses from which you can choose one or more option.
bold text	Bold text indicates emphasis.
<i>italicized text</i>	Italic type in text indicates the name of a referenced external document.
blue text	Blue text indicates either a hypertext link or an element name (in xml examples).
<code>monospaced text</code>	Used in code examples and elsewhere to designate field/element names.

Contact Information

This section provides contact information for organizations within Little & Co.

Implementation Support - For technical assistance to resolve issues encountered during the onboarding process, including LittleXML certification testing.

Implementation Contact Information

E-mail	implementation@litle.com
Hours Available	Monday – Friday, 8:00 A.M.– 6:30 P.M. EST

Chargebacks - For business-related issues and questions regarding financial transactions and documentation associated with chargeback cases, contact the Chargebacks Department.

Chargebacks Department Contact Information

Telephone	978-275-6500 (option 4)
E-mail	chargebacks@litle.com
Hours Available	Monday – Friday, 8:00 A.M.– 6:30 P.M. EST

Technical Publications - For questions or comments about this document, please address your feedback to the Technical Publications Department. All comments are welcome.

Technical Publications Contact Information

E-mail	TechPubs@litle.com
---------------	--

Relationship Management - For non-technical issues, including questions concerning the user interface, help with passwords, modifying merchant details, and changes to user account permissions, contact the Relationship Management Department.

Relationship Management Contact Information

Telephone	1-800-548-5326
E-mail	customerservice@litle.com
Hours Available	Monday – Friday, 8:00 A.M.– 6:30 P.M. EST



INTRODUCTION

The LitleXML data format supports two types of transaction submission methods – Online and Batch. With the Online method, you submit each transaction independently and receive a response in real-time. The Online method is used most often for Authorization and Void transactions. The Batch method enables you to submit multiple transactions simultaneously. Litle recommends the Batch method for all transaction submissions except Authorizations and Voids.

This chapter provides an overview of the LitleXML data format, including some basic XML coding requirements. Also discussed are the advantages of using batch processing for most of your transactional processing, duplicate transaction detection, report groups, and the various supported transaction types.

The topics discussed in this chapter are:

- [The LitleXML Data Format](#)
- [Batch Transaction Processing](#)
- [Duplicate Transaction Detection](#)
- [Coding for Report Groups](#)
- [Smart Authorization Features](#)
- [Card Filtering Services](#)
- [Authorization Recycling](#)
- [Tokenization Feature](#)
- [eCheck Processing](#)
- [Automatic Account Updater Service](#)
- [Healthcare Card Feature](#)
- [Supported Transaction Types](#)

1.1 The LitleXML Data Format

Although Litle & Co. supports transaction processing via many data formats, there are several advantages to using the LitleXML format. Some of the advantages are as follows:

- **Easier Implementation, Operations, and Debugging** - Compared to fixed length or binary formats, the XML format is considerably easier for operations staff to read and edit, using virtually any text editor. This allows Litle's Implementation, First Line Support, and Relationship Managers to quickly communicate any issues and work with your own operations staff to make necessary corrections without worrying about line lengths, padding or encoding.
- **Fewer Downgrades** - Since the LitleXML format allows you to explicitly tie deposits to their associated authorizations via the `<litleTxnId>` element, your transactions qualify for the best interchange rates at a higher frequency than with formats that do not support this transaction cross-referencing.
- **Simpler Capture (Deposit) and Refund Transactions** - Because the LitleXML format associates related transactions using the `<litleTxnId>` element, our format does not require you to resubmit all of the authorization information on a deposit nor all of the deposit information on a refund. When you submit the unique transaction id, Litle & Co. automatically pulls the information from the original transaction. Most other formats require you to resubmit the related data with each transaction.
- **Superior Reporting** - The LitleXML format allows you to separate your transactions into different categories by specifying a Report Group on each transaction. When accessing your data on the Litle Merchant Accounting System, this feature allows you to filter your financial reports by Report Groups, providing more granular detail based on a reporting hierarchy the Report Groups create. Most other formats restrict reporting categories to a batch or specific merchant id.
- **Improved Chargeback Management** - Unlike most other formats where transactional relationships can be a "best guess" proposition, the LitleXML format explicitly ties related transactions, allowing you and Litle to see authorization-to-deposit and deposit-to-refund relationships with precision. This knowledge is indispensable when fighting chargebacks.
- **New Features** - All new features introduced are first supported in the LitleXML format. In fact, some features may never be supported in other formats, since development is dependent upon a third party. For example, some features currently supported only in LitleXML are:
 - Verified by Visa and MasterCard SecureCode
 - Level III Data
 - Automatic Account Updater
 - Chargeback Management API
 - PayPal
 - Smart Authorization Response Data
 - Bill Me Later

1.1.1 Communications Protocols

There are four communication protocols supported for the submission of Batch transactions to Little & Co. for processing and one for Online submissions as shown in [Table 1-1](#). For Batch submissions Little recommends that you use one of the two FTP methods. Use the HTTPS Post method for Online transactions.

TABLE 1-1 Communication Protocol Support Matrix

Protocol	Encryption	Batch	Online
HTTPS Post	SSL	Supported	Recommended
FTP	PGP or GPG (open source)	Recommended	N/A
sFTP	SSH Key	Recommended	N/A

If you use the standard FTP method, you must use either the Pretty Good Privacy (PGP) or the open source GNU Privacy Guard (GPG) encryption for your submissions. Both of these encryption methods use key cryptography and support message authentication and integrity checking. The alternate method, Secure FTP (sFTP), uses Secure Shell (SSH) to secure the transmission.

1.1.2 General XML Coding Requirements

As part of the on-boarding process, you receive XML schema files from your Little Implementation Consultant. Using those files and this document as a guide, you create the required XML documents for submission of your transactions. You should validate all XML you create using the supplied schema. Also, working with your Little Implementation Consultant, you are required to perform various tests of your XML (see [Chapter 2, "Testing Your LittleXML Transactions"](#)) prior to submitting transactions to the production environment.

In addition to the process outlined above, there are a few XML basics of which you should be aware.

- Encode all data using the UTF-8 format.
- Although it is not required, Little recommends that when formatting your XML, you keep each element on its own line. This will aid in debugging situations where an error message specifies an issue in a particular line of XML code (for example, line 20).
- Be aware of special characters that require specific handling (see [Table 1-2](#)). For example, the less than (<) and greater than (>) symbols define element tags in the XML code. Using the entity names **<** and **>** instead of < and > prevents a browser from interpreting these characters as element brackets.

TABLE 1-2 Coding for Special Characters

Character	Description	Entity Reference (case sensitive)
<	less than	<
>	greater than	>
“	quotation	"
‘	apostrophe	'
&	ampersand	&

1.1.3 Other XML Resources

There are several Internet sites that provide both reference and educational information that may help you when implementing your XML. A few of these sites are:

- http://www.w3schools.com/xml/xml_syntax.asp
- <http://www.w3.org/>
- <http://www.utf-8.com/>

1.2 Batch Transaction Processing

Batch processing involves a group of transactions submitted in a single file. In the case of a LittleXML Batch the parent or root element is the `<littleRequest>` element. A single `<littleRequest>` can contain many batches and each batch can contain multiple transactions. Little & Co. recommends that you use Batch processing for all transaction types except Authorizations and Voids (Online only).

Some of the advantages of using Batch processing are:

- **Better Performance** - Little & Co. optimizes batch processing by processing multiple transactions in the batch simultaneously. This allows you to process thousands of transactions quickly without writing complicated logic or managing complicated processes.
- **Easier Reconciliation** - When processing a batch, all transactions within that batch post on the same day. In the case of Online transactions, you could submit two transactions at the same time and one could post today and the other tomorrow. This can cause confusion in your accounting process.
- **Easier Error Recovery** - A batch processes as a single unit, thus if you experience any system or communication issue while processing a batch, you can easily determine if the file was processed. With Online transactions, determining which individual transactions were not processed can be more difficult.

1.3 Duplicate Transaction Detection

Little & Co. performs duplicate transaction checking for both Online and Batch transaction submissions. This section discusses the different checking methodologies used depending upon the type of submission.

NOTE: For tokenized transactions, the token is used in place of the card numbers by the Duplicate Transaction Detection process.

For PayPal transactions a combination of the PayPal Id + the (consumer's) email is used by the Duplicate Transaction Detection process.

1.3.1 Batch Duplicate Checking

When processing a Batch file, the Little system acts to detect duplicate transactions for the following transaction types: Authorization, Auth Reversal, Capture, Force Capture, Capture Given Auth, Credit, Sales, eCheck Credit, and eCheck Sales.

For each of these transaction types, the application compares the transaction type, transaction amount, the `orderId` element from the request, credit card number, and credit card expiration date against transactions in other batch files processed within the previous five days. If the characteristics of the new transaction match a previously processed transaction, the system marks it as a duplicate.

IMPORTANT: If you do not include a value for the `id` attribute in the request, Duplicate Checking is not performed.

The system only performs duplicate detection against valid transactions from the previous five days. For example, if an Authorization request matches a declined Authorization from the previous day, the system would not count it as a duplicate, because the declined Authorization was not a valid Authorization.

Also, a Batch file must be processed completely to be included in the previous five days of data. For example, if multiple Batch files you submit are processing simultaneously, the system will not compare the transactions in one batch with the transactions in the other, because neither has completed processing. For this same reason the system cannot detect duplicate transactions within the same Batch file.

If the system detects ten consecutive duplicate transactions or if the number of duplicate transactions is greater than or equal to 25% of the total transactions in the batch, the system flags the batch file as a duplicate. In this case, Little will contact you so you can decide to discard the file as a duplicate or to continue processing the file.

1.3.2 Online Duplicate Checking

When processing an Online transaction, the Little system acts to detect duplicate transactions for the following transaction types: Capture, Force Capture, Capture Given Auth, Credit, Sales, eCheck Credit, eCheck Sales, eCheckVoid, and Void.

For most transactions the system compares the transaction type, the `id` attribute from the request, and the credit card number against other Online transactions processed within the previous two days. For transactions that reference other transactions (for example, a deposit referencing an authorization or a refund referencing a deposit), the system compares the transaction type, `id` attribute, and the card number from the referenced transaction (i.e. the transaction identified by the `<littleTxnId>` element) against other Online transactions processed within the previous two days.

IMPORTANT: If you do not include a value for the `id` attribute in the request, Duplicate Checking is not performed.

The system only performs duplicate detection against valid transactions. For example, if a Capture request matches a declined Capture from the previous day, the system would not count it as a duplicate, because the declined Capture was not a valid transaction.

NOTE: While it is uncommon, under certain circumstances network latency may cause a duplicate Sale transaction to go undetected as a duplicate. This can occur if you submit a second, duplicate Sale transaction while the response from the network for the Authorization portion of the first transaction is sufficiently delayed such that the first Sale has not been recorded as a valid transaction in the Little system.

If you elect to submit Online Sale transactions, Little recommends a timeout setting of not less than 60 seconds to reduce the chances of undetected duplicate Sale transactions.

If the system determines a transaction to be a duplicate, it returns the original response message with the `duplicate` attribute set to **true** (see example below). This attribute indicates that the response was returned on a previous transaction.

NOTE: If you do not receive a response for a submitted transaction, Little recommends you re-send the transaction. If Little received and processed the original transaction, you will receive a duplicate response. If Little did not receive the original transaction, the new submission will be processed.

Example: captureResponse for a duplicate Capture

```
<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <captureResponse id="2" duplicate="true" reportGroup="ABC Division">
```

```
<littleTxnId>1100030204</littleTxnId>  
<orderId>65347567</orderId>  
<response>000</response>  
<responseTime>2011-08-11T14:48:48</responseTime>  
<postDate>2011-08-11</postDate>  
<message>Approved</message>  
</captureResponse>  
</littleOnlineResponse>
```

1.4 Coding for Report Groups

You use Report groups (`reportGroup` attribute) to separate transactions from your merchantId into different categories so you can view the financial reports by your specific report group names. If you are unsure what groupings to use, your Relationship Manager can help you determine the best practice for your business.

NOTE: The `reportGroup` attribute is case and space sensitive. A `reportGroup = "Picture Frame"` is a different report group than a `reportGroup = "pictureframe"`.

Example: Report Groups for Two Products

Company A sells two products: picture frames and coffee mugs. To view the financial information for each product, the merchant should code their transactions such that a picture frame transaction uses `reportGroup = "Picture Frame"` and a coffee mug transaction uses `reportGroup = "Coffee Mug"`. This allows you to report as nested levels: Company A (the main/top level), or Picture Frame and Coffee Mug (individual report group).

The online reporting UI would show the information from this scenario as follows:

Company A

Picture Frame

Coffee Mug

Example: Report Groups for Two Products with Sub-groups for Order Method

Here is an example of the three levels of reporting. Company B sells two products: picture frames and coffee mugs. In addition, they have two methods of taking orders: over the phone, or via a Website. They want to structure their reports to provide information on each product, as well as the order method.

Working with their Little Implementation Consultant, the merchant arranges for two rollups to be included in their report hierarchy: Picture Frame Rollup and Coffee Mug Rollup. Each rollup will contain nested reports for the Product/order method. The merchant then codes the `reportGroup` attribute in the appropriate manner to reflect the nesting.

For example, any transaction for a picture frame taken over the phone could be coded with a `reportGroup = "Picture/Phone"` and orders taken via the Web could be coded with a `reportGroup = "Picture/Web"`. The resulting structure would have 3 levels. The main/top level (Company B), each rollup (Picture Frame or Coffee Mug rollup), and each individual report group (Picture/Phone, Picture/Web, Mugs/Phone, and Mugs/Web).

Assuming the same basic structure was used for the Mugs product, the online reporting UI would show the information from the scenario as follows:

Company B

Picture Frame Rollup

Picture/Phone

Picture/Web

Coffee Mug Rollup

Mugs/Phone

Mugs/Web

1.5 Smart Authorization Features

The Smart Authorization set of features are designed provide additional information to merchants, allowing them to improve authorization approval rates and lower the total cost of payments. Currently, Little & Co. offers six features in the Smart Authorization family of features: Prepaid Indicator, Affluence Indicator, Issuer Country Indicator, Card Type Indicator, Prepaid Card Filtering, and International Card Filtering.

NOTE: Both features described in this section require activation. Please consult your Relationship Manager for additional information.

1.5.1 Prepaid Indicator

Studies show that branded prepaid cards are growing in popularity with consumers. These cards are available in the form of non-reloadable Gift cards, Consumer Rebate/Incentive cards, and Teen cards among others. The Prepaid Indicator feature acts to determine if the submitted card is a prepaid card. If so, the system returns the `type` element with a value of Prepaid and the `availableBalance` element stating the outstanding balance remaining on the card (if available).

Knowing that the card is prepaid, as well as the available balance, at the time of sale is especially useful for merchants engaged in recurring payment, installment payment, or deferred billing scenarios. Merchants in these situations can use the information made available by this feature to make intelligent decisions concerning the profitable management of prepaid card usage by avoiding several factors that may contribute to lost revenue, while taking advantage of other opportunities that may add to revenue and enhance the customer experience.

For example, one possible situation merchant can avoid is fraudulent deferred/installment payment purchases made with a prepaid card that does not have enough available balance to cover the subsequent payments. With the available balance known, merchants can determine if the card can meet the required payment structure. If the card's balance does not meet the required threshold, the merchant can request another payment method, which may result in eliminating fraudsters, while retaining legitimate customers.

Another more common situation occurs when the consumer is unaware of the card balance. If the transaction is rejected due to inadequate balance, perhaps repeatedly, it could result in an unsatisfied customer and an abandoned purchase. Alternately, the card could have slightly more than the required balance, which the consumer would spend, if they had the knowledge. If the available balance is insufficient for the purchase, the merchant can obtain a second or alternate payment method. If the balance is higher than required for the purchase, the merchant may be able to encourage additional purchases.

In addition to indicating if the submitted card is a prepaid card and the available balance, this feature includes information about whether the card is reloadable and the specific type of prepaid

card (i.e., TEEN, GIFT, PAYROLL, etc.). You can use this information to further refine your sales and marketing strategies.

1.5.2 Affluence Indicator

Visa, MasterCard, and Discover provide enhanced credit card products for consumers with high disposable incomes and high card spending. These cards encourage usage by offering the cardholders additional benefits usually including reward incentives, no pre-set spending limit, higher authorization approval rates, faster access to a customer service representatives, and dedicated chargeback resolution support.

Litle & Co. analysis of payments data indicates that consumers using these cards types typically spend more per order than consumers using traditional credit and debit cards. The Affluence Indicator feature provides the ability for merchants to segment their consumers based on the affluence level as determined by the issuer. Within the LitleXML Authorization response, consumers using these enhanced card products are classified either as Mass Affluent or Affluent. Based upon the specific card type, high income consumers are classified as Mass Affluent, while high income-high spending consumers are classified as Affluent.

Having this information at the time of authorization, allows merchants the opportunity to adjust their sales approach to the needs and spending patterns of the consumer, potentially generating additional sales. Having this information on file for later analysis also may provide the opportunity for targeted marketing campaigns and future sales.

1.5.3 Issuer Country Indicator

Knowing the country of the Issuing bank helps you in two respects. From a sales and marketing standpoint, this knowledge allows you to better analyze the purchasing patterns of your customers based upon their country of origin. You can then tailor marketing campaigns to take advantage of this geographic information. Likewise, you can use this information to analyze the successfulness of tailored campaigns.

The second advantage to having this information readily available, is that you can use it to help determine possible patterns of fraud. With this knowledge in hand you can use the Litle International Card Filtering feature to limit your exposure to international fraud originating in particular geographic locations.

1.5.4 Cardholder Type Indicator

The Card Holder Type indicator is an additional data point Litle & Co. can provide as part of the Smart Authorization family of features. This indicator returns whether the submitted card is a commercial or consumer card, providing you with additional data useful when analyzing sales patterns and/or planning marketing campaigns.

1.6 Card Filtering Services

As part of the Smart Authorization suite of features, Litle & Co. currently offers two types of card filtering services that may aid you in reducing certain types of fraud: Prepaid Card Filtering and International Card Filtering.

1.6.1 Prepaid Card Filtering Service

Just because a credit card network/company returns a valid authorization for a purchase does not always mean that completing the transaction is in your best interest. There are several reasons you may wish to decline a sale on a particular card at a particular time. Many merchants engaged in recurring payment, installment payment, or deferred billing experience some loss due to fraud schemes that make use of prepaid cards. Consider the case of a consumer using a prepaid card with a balance of \$100 to make a purchase that involves an initial charge of \$50 followed by three installments of \$50 each. The authorization would be approved for the initial transaction, and the card might have adequate balance for an additional charge, but if the consumer was attempting to defraud the merchant or simply used the card for other purchases, the card may not have sufficient balance for any additional payments. While the Prepaid Indicator feature provides you with the information necessary to make a decision at the time of the sale, and to request a secondary or different payment method, instead you may wish to have Litle filter these transactions automatically when you send the Authorization transaction.

If you elect to use the Litle & Co. Prepaid Card Filtering Service, you can select one of two methods of implementation. Using the first filtering method, our system declines all Authorization and Sale transactions when the consumer uses a prepaid card. In this case, if you are using LitleXML schema version 8.3 or above, the system returns a Response Reason Code of **309 - Restricted Card - Prepaid Card Filtering Service**. If you use LitleXML version 8.2 or below, the system returns a Response Reason Code of **322 - Invalid Transaction**. This method also allows you to disable the filtering logic on a transactional basis by including the `<prepaid>` element set to a value of **false**, thus allowing you to accept a prepaid card for these transactions.

The second method of implementing the Prepaid Card Filtering Service is to use it only on selected transactions. To enable the filter on a particular transaction, set the `<prepaid>` element set to a value of **true**. This method would be useful to a merchant who offers products with both one-time payments and installment payments. For products involving a single payment, you may want to allow the use of prepaid cards, while for the product with multiple payments you may want to filter the use of prepaid cards.

NOTE: Within either implementation method, you can elect to filter all prepaid cards, only reloadable prepaid cards, or only non-reloadable prepaid cards. Please consult your Litle Implementation Consultant for additional information about setting these global parameters.

1.6.2 International Card Filtering Service

An examination of your historical fraud data may show a high percentage of fraudulent transactions originating with certain international cards. You may wish to limit your exposure to this type of fraud by taking advantage of the Litle & Co. International Card Filtering Service. This feature allows you to filter cards originating in either all foreign countries or selected foreign countries based upon the country of the card issuer (filtering based upon individual country is available only for Visa and MasterCard).

If you elect to use this feature, when you submit an Authorization/Sale transaction, the Litle system determines the country of origin of the card. If the card originates outside the United States and you have elected to filter all international cards, the system declines the transaction. Likewise, if you have elected to filter a specific country or countries and the card originates from a designated country, the system declines the transaction. If you are using LitleXML schema version 8.3 or above, the system returns a Response Reason Code of **312 - Restricted Card - International Card Filtering Service**. If you use LitleXML version 8.2 or below, the system returns a Response Reason Code of **322 - Invalid Transaction**.

As with the Prepaid Card Filter Service, you can override your settings on a transactional basis by including the `<international>` element set to **false** when you submit the Authorization/Sale transaction. In this case, the system ignores the filtering service and processes the transaction normally.

1.6.3 Chargeback Filtering

If you elect to use the Chargeback Filter Service, the Litle system checks your historical data to see if you have received a chargeback from the same account within the last 90 days. If you did, the system filters the Authorization/Sale and returns a Response Reason Code of 308 - Restricted Card - Chargeback.

As with the other filters, you can override the filter on a per transaction basis by including the `<chargeback>` element set to **false** when you submit the Authorization/Sale transaction. In this case, the system ignores the filtering service and processes the transaction normally.

1.7 Authorization Recycling

Authorization recycling is the process of retrying declined authorization attempts. Every merchant, especially those with a business model that uses recurring or installment payments, must determine a strategy for dealing with declined authorizations. As part of optimizing their operations, merchants must devise plans for both the frequency and number of recycling attempts before contacting the cardholder or risking interruption of service. If a merchant does not recycle enough, they risk losing customers and revenue; whereas, if the merchant recycles too often, they risk increasing their total cost of payments. Implementing an optimal recycling strategy aids customer retention and therefore yields higher revenues, while lowering the costs of payment acceptance and improving cash flow.

Little & Co. offers two options to help you with your recycling, Recycling Engine and Recycling Advice. Both offering have the following benefits:

- Increases approval rates
- Shortens time to approval, improving cash flow
- Reduces the number of authorization retries
- Lowers the risk of account/service cancellation

1.7.1 Recycling Engine

The Little Recycling Engine is a managed service that automatically retries declined authorization attempts on your behalf. It requires little or no IT investment on your part. Also, implementing the Little service removes the need to plan your own recycling strategy.

In order to determine the most effective recycle timing, Little performs statistical analysis of past recycling attempts across our entire merchant portfolio. This analysis examines many factors, including method of payment, response codes, and transaction amount among others, to determine the optimum intervals between attempts to obtain a successful authorization. When you receive a declined Authorization, the Little system automatically queues the transaction for a retry at a designated time. Recycling of the Auth continues until it is either successful or the algorithm determines that it is no longer advantageous to retry.

NOTE: For Visa transactions, the recycling Engine will retry declined Authorizations a maximum of 4 times within 16 days, as limited by Visa regulations.

Also, the Recycling Engine will not recycle transactions declined with response code 328 - Cardholder requested that recurring or installment payment be stopped.

Little provides the results of the recycling efforts to you in a batch file posted daily to an FTP site. This file contains transactions that either approved or exhausted the recycling pattern on the

previous day. If you submit an Authorization for a transaction in the recycling queue, Little returns the response from the last automatic recycling attempt. To halt recycling of a particular transaction, submit either an Authorization reversal transaction, if the original transaction was an Auth, or a Void transaction, if the original transaction was a Sale (conditional deposit).

1.7.2 Recycling Advice

If you choose to schedule and re-attempt failed authorizations on your system, Little & Co. can provide recycling advice designed to optimize the number of attempts needed to receive an approved authorization. In this case, Little performs the same data mining as with the Recycling Engine the difference being that merchants utilizing this feature receive the advice as part of the response message for declined Authorization or Sale requests. You must then schedule the next authorization attempt using the advice provided.

NOTE: You must upgrade to LittleXML schema V8.6 or above to receive Recycling Advice.

1.7.3 Authorization Recycling Best Practices

In order for Little to provide optimum recycling advice, merchants using this feature should adhere the following best practices:

- The recycled transactions should maintain the identical information as the initial declined transaction for the following fields/elements and attributes:
 - Authentication information - <user> and <password>
 - Report group attribute - reportGroup = "UI Report Group"
 - Order Id - <orderId>
 - Amount - <amount>
 - Card data (or token data) - <type> and <number> (or <type> and <littleToken>)
- If you are a recurring merchants who charges the same card/same amount each billing cycle, use a new <orderId> at the start of the billing cycle, but keep the same <orderId> when recycling a charge.
- If different entities within your organization use different billing cycles, each should have its own merchant identifier (MID).
- Inform your Little Relationship Manager of your billing cycle (in days) and any updates to it.

1.8 Tokenization Feature

Tokenization is the process by which a credit card number or eCheck account number is replaced by a reference number, referred to as a token. Unlike the card or account number, you can store the token on your system without concern of a security breach exposing critical customer information. Litle & Co. stores the information in a secure vault and accesses it only when you submit a transaction using the supplied token.

NOTE: You must be enabled for card tokenization in order to use the eCheck tokenization feature.

In the case of credit cards, since you do not store the customer's account information, the scope of PCI requirements to which you must comply may be minimized. This may greatly reduce the cost of compliance and may limit your liability if your systems are breached.

NOTE: Litle & Co. recommends you consult your own PCI Compliance and Legal departments to determine the specific advantages of tokenization for your company.

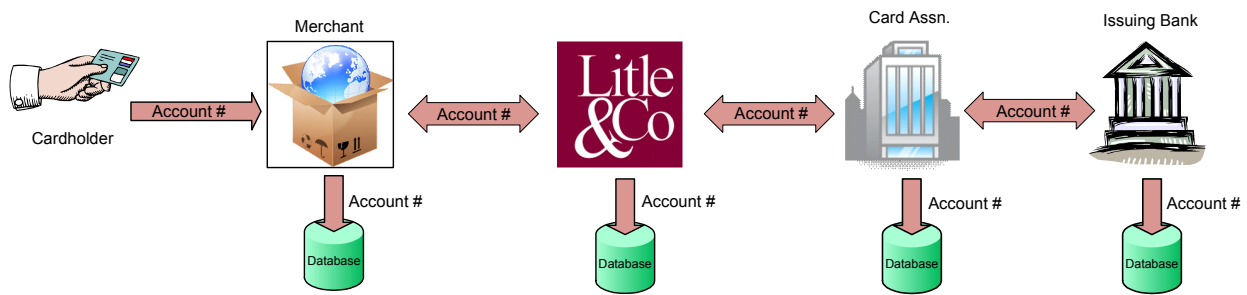
This section discusses the following topics:

- [How Tokenization Works](#)
- [Litle Token Formats](#)
- [Obtaining Tokens](#)
- [Supported Token Transactions](#)

NOTE: Information about the use and integration of the Litle Pay Page is contained in the Litle & Co. Pay Page Integration Guide.

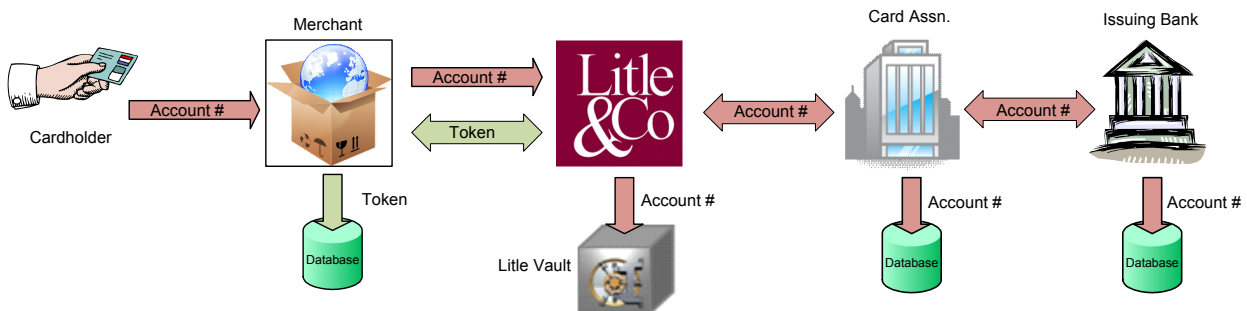
1.8.1 How Tokenization Works

In a non-tokenized environment, customer data, including the card/eCheck account number, is handled and stored by multiple parties for each transaction. From a merchant standpoint, they receive the information, store it in their own database, and transmit it to their processor with the transaction request, as [Figure 1-1](#) shows for card information. While the access and transmission of the data may occur a single time, as in the case of a Sale transaction, frequently the data is transmitted multiple times in order to complete a single sale, as in the case of an Auth followed by a Capture or several partial Captures. The local storage and repeated transmission of the information creates additional possible breach points, where the information might be compromised by a malicious third party.

FIGURE 1-1 Card Information Flow in Non-Token Environment

In a tokenized environment customer data is ideally transmitted a single time and is never stored locally by the merchant, as Figure 1-2 shows for card data. Once the account number is registered, using either a `registerTokenRequest` or by submitting it with any supported transaction, Litle returns a token. You store the token locally and use it for all future transactions concerning that account. Litle takes responsibility for storing and safeguarding the account information.

NOTE: The difference between card data flow and eCheck data flow is that the entities upstream of Litle & Co. are different. The principles remain the same from a merchant standpoint.

FIGURE 1-2 Card Information Flow in Tokenized Environment

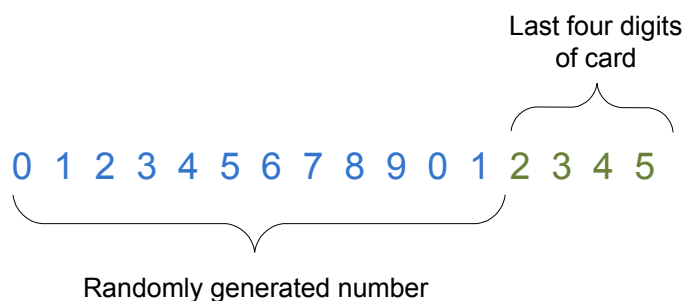
NOTE: Depending upon implementation, the use of a pay page can allow the account information to come directly to Litle, so the merchant handles the token only.

1.8.2 Litle Token Formats

For credit cards, in an effort to minimize development requirements on the merchant side, Litle & Co. elected to use a format-preserving tokenization scheme. In simple terms this means that the length of the original card number is reflected in the token, so a submitted 16-digit number results in a 16-digit token. Also, all tokens use only numeric characters, so you do not have to change your systems to accept alpha-numeric characters.

The credit card token numbers themselves have two parts. The last four digits match the last four digits of the card number. The remaining digits (length can vary based upon original card number length) are a randomly generated. Unlike credit card numbers, which are Mod 10 compliant, tokens are Mod 10 + 1 compliant.

FIGURE 1-3 Token Format - Card



For an eCheck token, since the account number length can vary widely, Litle & Co. elected to make the tokens a uniform length of 17 digits. Unlike card tokens, the entire eCheck token number is a randomly generated. The system supplies the last three characters of the account number in a separate element. As with credit card tokens, eCheck tokens are Mod 10 + 1 compliant.

1.8.3 Obtaining Tokens

NOTE: You must be token enabled and certified prior to using the Vault feature. Please consult your Litle Relationship Manager concerning the requirements and details of this process.

There are three ways for you to obtain tokens for a account numbers. First, you can submit an existing card number/eCheck account information (account number and routing number) using a Register Token request. When Litle receives this transaction type, we generate a token and return it to you via a Register Token response (see [Register Token Transactions](#) on page 173.) Although you can use this method to tokenize an account number at any time, it is most useful when initially tokenizing your customer database. Litle & Co. recommends that you collect all distinct credit card numbers in your database and submit the information in one or more large batch files. When you receive the response file, parse the returned token information to your database, replacing the card numbers.

The second method you can use to obtain a token is to submit a supported transaction with the card information. If you are a tokenized merchant, Litle will automatically convert the submitted card number to a token and return it to you in the transaction response. Typically, you would use this method when taking and submitting a transaction during the normal course of business. When you receive the response, you store the token instead of the card information.

NOTE: Once a card number has been converted to a token for a particular merchant, subsequent submissions of the same card number will return the same token.

The third method of obtaining a token applies only to merchants using the Litle Pay Page feature. In this case, upon submission of an account number via the Pay Page API, Litle issues a Registration Id. You then submit the Registration Id in an Authorization or Sale transaction and receive the token in the response message.

1.8.3.1 Bulk Token Registration

If you are new to Litle & Co., and have utilized tokens with a previous processor, Litle can perform a bulk token registration on all the card numbers that were vaulted with your previous processor. Litle will request an encrypted file containing card and token information directly from your previous processor, thereby maintaining PCI compliance. The process is as follows:

1. During your implementation with Litle & Co., a Litle Implementation Consultant contacts your previous processor and requests an encrypted batch file containing card and token numbers for your customers. The file can be comma-delimited, tab-delimited, or any other common format.
2. Litle performs a bulk token registration of all of the card numbers contained in the file.
3. Litle returns a file to your organization containing the old tokens and the new Litle-issued tokens, so that you can update your order processing system.

Note that Litle & Co. supports token-extractor formats of all major token service providers. Contact your Litle Implementation Consultant for more information or to initiate this process.

1.8.4 Supported Token Transactions

The following transactions support the generation and use of tokens:

- **Authorization** - You can submit the transaction either with a token or card information. If you submit card information, Litle automatically generates the token and returns it in the response.
- **Capture Given Auth** - You can submit the transaction either with a token or card information. If you submit card information, Litle automatically generates the token and returns it in the response.

- **Credit** - You can submit the transaction either with a token or card information. If you submit card information, Litle automatically generates the token and returns it in the response.
- **Force Capture** - You can submit the transaction either with a token or card information. If you submit card information, Litle automatically generates the token and returns it in the response.
- **Register Token** - You use this transaction to convert a card number or eCheck account number to a Litle token without an associated authorization, verification or payment transaction.
- **Sale** - You can submit the transaction either with a token or card information. If you submit card information, Litle automatically generates the token and returns it in the response.
- **eCheck Credit** - You can submit the transaction either with a token or account information. If you submit account information, Litle automatically generates the token and returns it in the response.
- **eCheck Redeposit** - You can submit the transaction either with a token or account information. If you submit account information, Litle automatically generates the token and returns it in the response.
- **eCheck Sale** - You can submit the transaction either with a token or account information. If you submit account information, Litle automatically generates the token and returns it in the response.
- **eCheck Verification** - You can submit the transaction either with a token or account information. If you submit account information, Litle automatically generates the token and returns it in the response.

1.8.5 Compliance with Visa Best Practices for Tokenization

As shown below, the Litle Vault tokenization solution complies with 11 of the 12 items listed in the Visa Best Practices for Tokenization document. The twelfth item concerns the management of stored historical data (that may contain card information) within your systems. Tokenizing all historical card info when implementing the Litle solution would satisfy this item, as would protecting it per PCI DSS requirements.

TABLE 1-3 Visa Best Practices for Tokenization Compliance

Item #	Who	Domain	Best Practice	Complies?
1	Litle	Tokenization System	Network Segmentation	Yes
2	Litle	Tokenization System	Authentication	Yes
3	Litle	Tokenization System	Monitoring	Yes
4	Litle	Tokenization System	Token Distinguishability	Yes
5	Litle	Token Generation	Token Generation	Yes

TABLE 1-3 Visa Best Practices for Tokenization Compliance

Item #	Who	Domain	Best Practice	Complies?
6	Little	Token Generation	Single- vs. Multi- use Tokens	Yes
7	Little	Token Mapping	PAN Processing	Yes
8	Little	Card Data Vault	PAN Encrypted in Storage	Yes
9	Little	Card Data Vault	Covered by PCI DSS	Yes
10	Little	Cryptographic Keys	Key Strength	Yes
11	Little	Cryptographic Keys	Covered by PCI DSS	Yes
12	Merchant	Historical Data Management	Non-tokenized data protected	Merchant Implementation Decision

1.9 eCheck Processing

An eCheck is an alternative payment method that directly debits a consumer's account via the Automatic Clearing House (ACH) network. From a merchant's standpoint offering eCheck as a payment method has several advantages, including a large consumer base in excess of 130 million accounts and no interchange fees.

This section provides information about several Little & Co. eCheck processing features. Please consult with your Relationship Manager for additional information.

NOTE: eCheck Void transactions are supported only as Online transactions.

1.9.1 Validation Feature

Little & Co. performs a validation of the eCheck routing number. This is done both to verify that the routing number is correctly formatted and that it exist in the Fed database. If the routing number fails this validation, the transaction is rejected. Little & Co. performs this validation on all eCheck transactions automatically.

1.9.2 Verification Feature

Since there is no authorization process associated with eChecks allowing you to confirm the availability of funds and hold the purchase amount, there is a higher risk of certain types of fraud. The optional eCheck Verification feature allows you to submit an eCheck account number for comparison to a database containing historical information about the account, as well as the account holder. When you submit an eCheck Verification transaction the information you provide is compared to a negative database to see if the account is associated with activities, such as fraud, over drafts, or other items determined to be risk factors.

NOTE: Little & Co. makes use of a third party service, Certegy Check Services Inc., for all verification operations.

You can also initiate an account verification operation as part of an eCheck Sale transaction by setting the <verify> element to **true**. In this case, the eCheck Sale transaction is conditional upon the verification passing. If the verification fails, the sale is not processed.

1.9.2.1 Required Contents of Decline Notice

In the event you elect to perform verification on a transaction and also elect not to proceed with the transaction based upon a verification failure, you must provide your customer with the following Decline Notice. You can provide the notice orally, electronically, via e-mail, and/or via

U.S Mail, depending upon the type of transaction. The notice must be substantially as the notice set forth below that contains the disclosures required under the Fair Credit Reporting Act and instructs your customer how to contact Certegy directly.

NOTE: If the required language of the Decline Notice changes, Little & Co. will notify you of the change. You must enact the changes within 10 days.

Example: Decline Notice

We're sorry, but we are unable to proceed with your transaction. This determination was based on information provided by Certegy Check Services, Inc. ("Certegy"). To protect your privacy, Certegy did not provide any financial information to [Client's Name] during the authorization process.

The reason your transaction was not authorized was due to [mark one of the following based on applicable decline code transmitted by Certegy]:

- account closed
- dishonored check or transfer information contained in Certegy's files
- Certegy had insufficient information available
- the identification information you entered did not conform to established guidelines

You have the right under the Fair Credit Reporting Act to know the information Certegy utilized to make a determination regarding your check. If you find that any information Certegy utilized in its decision is inaccurate or incomplete, you have a right to dispute it with Certegy.

You may call Certegy toll free at 800-695-1854, or write to Certegy Check Services, Inc., P.O. Box 30046, Tampa, FL 33680-3046.

If you contact Certegy, please provide the following information so they can respond promptly to your request:

- | | |
|-------------------------------|-----------------------------------|
| • First Name | • Driver's License Number & State |
| • Current Address | • Home Telephone Number |
| • Date Declined | • Date of Birth |
| • Dollar Amount | • Social Security Number |
| • Check/Draft/Transfer Number | • Merchant Name |
| • Checking Account Number | • Name of Financial Institution |

1.9.3 Automatic Notice of Change (NoC) Updates

Similar to an issuing bank providing credit card Account Updater information, RDFIs provide Notification of Change (NoC) files and deliver them through the ACH network. These NoCs include updated account information including bank routing numbers, account numbers, and account names.

Litle & Co. makes available the NoC information to you for your use in updating your customer files. Additionally, if you submit a transaction containing information that has changed, we automatically update the information and forward the corrected transaction to the ACH network. The LitleXML response message to you also contains the updated information for your use in correcting your database.

1.9.4 Auto Redeposit Feature

NACHA rules allow merchants to redeposit entries when the initial deposit was returned for either Insufficient Funds or Uncollected Funds. Two redeposit attempts are allowed within 180 days of the settlement date of the initial deposit. Litle & Co. offers an optional service that allows you to preconfigure automatic redeposits of transactions returned for the those reasons. You define the number of days from the initial return for Litle to resubmit the transaction. You also define the number of days from the return of the first resubmission for the attempt of a second resubmission.

NOTE: You track the current state of your transactions, returns, and resubmissions via the Litle Merchant Accounting System User Interface. Please refer to the *Litle & Co. User Interface Guide* for additional information.

For example, you submitted an eCheck Sale transaction on 29 January that is returned for Return Reason Code R01 - Insufficient Funds. The return occurs on 1 February. With the Auto Redeposit feature enabled and a preset period of 5 days for the first redeposit, the Litle system would automatically generate a resubmission of the deposit on 6 February. If this transaction is also returned for the same reason code on 7 February and you have a preset time period for the second redeposit on 7 days, the Litle system generates the second redeposit on 14 February.

1.10 Automatic Account Updater Service

Credit and debit card numbers change for a variety of reasons including card expirations, card product type upgrades, portfolio conversions, and compromised account numbers among others. For merchants who offer services that are billed on a recurring or installment basis (for example, web hosting, gym memberships, specialized social networking, career services, monthly donation plans, etc.) out-of-date payment information can result in lost revenue, involuntary churn and decreased customer satisfaction.

Prior to the development of the Automatic Account Updater service, the standard method for merchants to obtain updated account information was to submit a batch file containing existing card information, requesting that Litle check for updates. Typically, merchants request updates for customer accounts scheduled to be billed in the next billing cycle. This legacy method is a relatively slow process, requiring several days for Litle to accumulate responses from the card networks/issuers and then to make the response file available to the merchant. Merchants must then update their billing systems with the new information, requiring IT processing cost. Failure to update their files can result in multiple requests (and charges) for the update information, as well as delays in or lost revenue, higher Authorization expenses, and possibly chargebacks when old account information is used.

With the new Litle & Co. Automatic Account Updater service you can shift the workload of obtaining and maintaining updated account information to Litle. Utilizing configurable scheduling algorithms, Litle initiates account update requests on your behalf and then stores the updated card information in its systems, effectively eliminating the standard file-based process completely. Once updated information is stored in our database, you submit billing transactions normally and Litle repairs the card information before submitting it to the networks for authorization.

This section discusses the following topics:

- [Automatic Account Updater Service Options](#)
- [Merchant Requirements](#)
- [Automatic Account Updater Features](#)

1.10.1 Automatic Account Updater Service Options

With a traditional account updater implementation, you submit update requests. These requests are then parsed into separate groups based upon the card type and forwarded to Visa, MasterCard, and Discover as required. When all responses have returned from the card networks, a response file is created and made available to you. The entire cycle from submission to availability of the response file may take up to five days. Once you have the update information, you can update your database and (re-)submit the transaction using the new account information. From creating queries that search your database for cards approaching their billing dates, to the coding required to parse the returned information correctly and update your databases, a significant investment of resources and time on your part is required.

The two Automatic Account Updater options are:

- [Full Service Option](#)
- [Match and Repair Option](#)

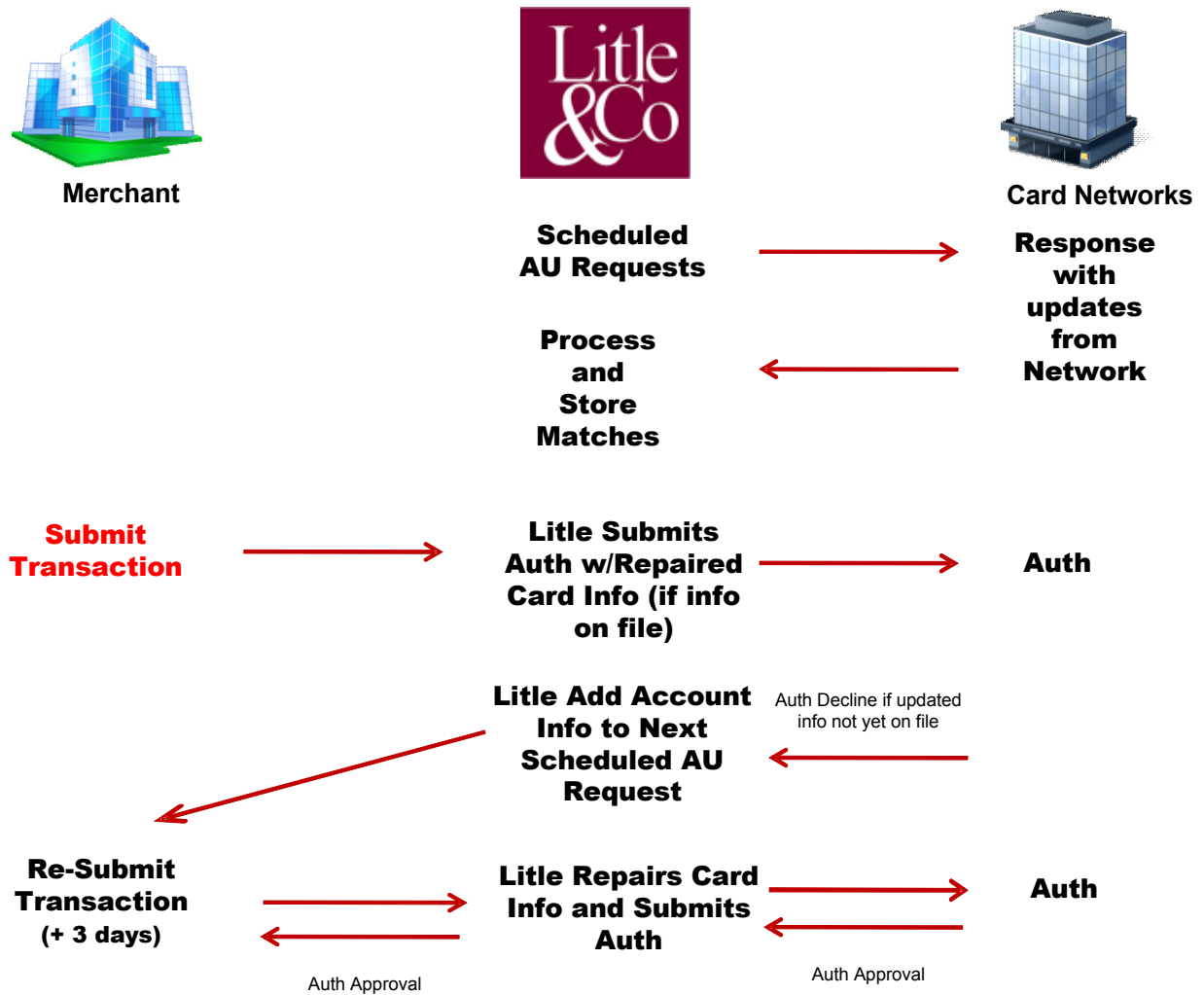
NOTE: It is always a best practice to link transactions through the `littleTxnId` element whenever possible. This is especially important when using Automatic Account Updater and the account number has changed. For example, if you submit a Sale transaction using account number A and once AAU repairs it to use account number B, the transaction is accepted. A week later you submit an orphan refund (does not reference the `littleTxnId`) using account number A. The refund may not get applied to the correct account, because there was no repair, and a chargeback may ultimately occur.

1.10.1.1 Full Service Option

If you select the Full Service option (see [Figure 1-4](#)), Little automatically initiates account update queries to the card networks on your behalf, utilizing configurable scheduling algorithms and rules, including the occurrence of declined Authorizations. Little records and stores any matches returned by the networks in a merchant specific database.

When you submit a transaction, our system checks the submitted card data against our database of updated information and repairs the information if needed. Since with this option Little does not transmit the update information to you, you can continue to submit transactions using the original card information and Little take responsibility for maintaining the updates. There is no requirement for you to ever update the card information in your system.

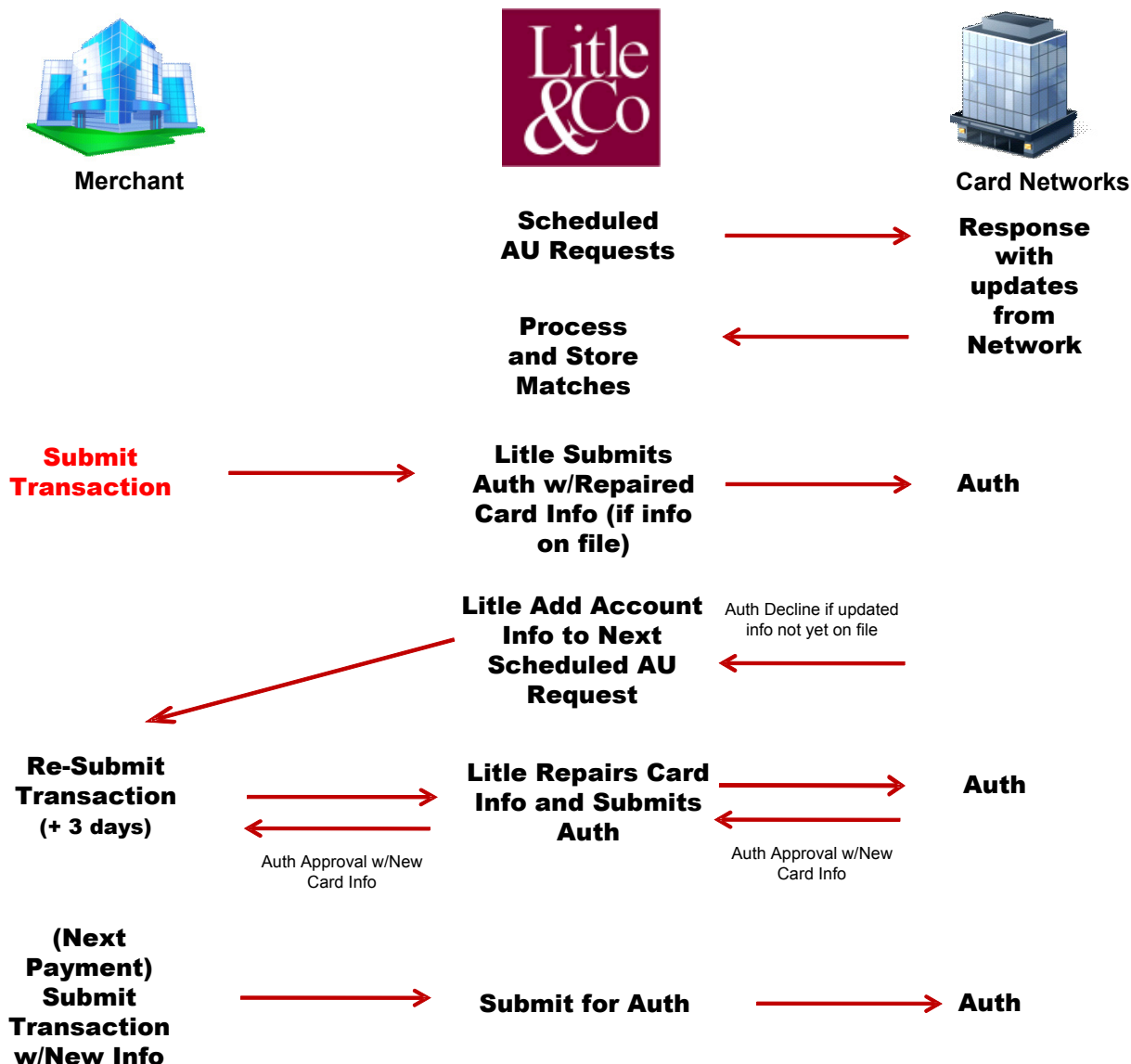
FIGURE 1-4 Full Service Option



1.10.1.2 Match and Repair Option

With the Match and Repair Automatic Account Updater option (see [Figure 1-5](#)), Litle performs all actions described in the Full Service option, but also returns the updated information to you in the LitleXML response messages. This allows you to update your database with the new information, using the repair service as a fallback, such as for cases where subsequent purchase by the same consumer occurs prior to you updating the account information in your database.

FIGURE 1-5 Match and Repair Option



1.10.2 Merchant Requirements

In order to use the Litle Automatic Account Updater service, you must first apply for membership to the following:

- MasterCard Automatic Billing Updater
- Visa Account Updater
- Discover Account Updater (not required by Discover for Litle acquired merchants)

Your Automatic Account Updater Welcome Kit includes the required application forms. If you have any questions about these forms, contact your Litle Relationship Manager, who can walk you through the application process. Approval from Visa and MasterCard typically takes between 10-15 business days. Normally, merchants are approved without issue; however, you can be declined for a variety of reasons. For example, merchants on a risk mitigation program typically are not accepted.

NOTE: Visa does not allow merchants with SIC numbers 5962, 5966, 5967, or 7995 to participate in their Account Updater service. MasterCard has no restrictions against any specific MCC numbers

1.10.3 Automatic Account Updater Features

The Litle Automatic Account Updater service can include the following features depending upon the implementation option you select:

- Litle initiates requests for updated account information to card networks based upon your billing cycle.
- Litle initiates requests for updated account information following certain failed Authorization attempts.
- All updated card information stored (per merchant) in our secure database.
- Automatic repair/replacement of outdated information with updated information in new Authorization/Sale transaction submissions.
- Return of the updated account information in the LitleXML response message when auto-repair occurs.
- Maintenance of card information history, so that the system can repair a card even if multiple updates have occurred during the card's billing lifecycle.
- All linked (to an Authorization) transactions will use the updated account information from the repaired parent transaction, including Captures, Refunds, and Reversals. If a re-Auth is needed on an attempted capture due to an expired authorization, the system uses the updated account information.
- Integration with Litle Vault for merchants utilizing Litle's tokenization solution.
- Return of Extended Response Codes in the LitleXML response messages.

1.11 Healthcare Card Feature

Today, there are several types of Healthcare accounts that allow participants to use pre-tax money for the purchase of IRS approved healthcare products and services, such as prescription medications and office visit payments/co-pays. The most common of these accounts are Flexible Spending Accounts (FSAs), Health Reimbursement Arrangements (HRAs), and Health Savings Accounts (HSAs). In order to provide consumers with a more convenient method of making use of these accounts, certain issuers provide signature based, MasterCard or Visa branded, healthcare payment debit cards.

To facilitate the processing of transactions related to these cards, Little & Co. has augmented the LittleXML format with elements specific to Healthcare card purchases. The `healthcareIIAS` element has been added to the Authorization and (Conditional) Sale transaction types. You use this element and its children to detail the costs associated with Healthcare related, IIAS approved items purchased by the consumer. The example below shows an Authorization transaction with IIAS items.

Example: Authorization with healthcareIIAS Element

NOTE: The example below includes the `visionAmount` and `dentalAmount` elements to show all available amount classifications. Since these are optional elements and have values of 0 in the example, they can be omitted.

```
<littleOnlineRequest version="8.2" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>Merchant1</user>
    <password>Password</password>
  </authentication>
  <authorization id="834262" reportGroup="ABC Division" customerId="038945">
    <orderId>123456789</orderId>
    <amount>5500</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
      <name>John Smith</name>
      <addressLine1>100 Main St</addressLine1>
      <city>Boston</city>
      <state>MA</state>
      <zip>12345</zip>
      <email>jsmith@someaddress.com</email>
      <phone>555-123-4567</phone>
    </billToAddress>
    <card>
      <type>VI</type>
```

```

    <number>4000000000000001</number>
    <expDate>1211</expDate>
    <cardValidationNum>555</cardValidationNum>
  </card>
  <cardholderAuthentication>
    <authenticationValue>BwABBjQ1gJDUCAAAAAA= </authenticationValue>
    <authenticationTransactionId>gMV75TmjAgk= </authenticationTransactionId>
  </cardholderAuthentication>
  <allowPartialAuth>true</allowPartialAuth>
  <healthcareIIAS>
    <healthcareAmounts>
      <totalHealthcareAmount>5500</totalHealthcareAmount>
      <RxAmount>4000</RxAmount>
      <visionAmount>0</visionAmount>
      <clinicOtherAmount>1500</clinicOtherAmount>
      <dentalAmount>0</dentalAmount>
    </healthcareAmounts>
    <IIASFlag>Y</IIASFlag>
  </healthcareIIAS>
</authorization>
</littleOnlineRequest>

```

Please keep in mind the additional following requirements/recommendations:

IMPORTANT: The information below is not intended as an exhaustive list of the requirements for the acceptance of Healthcare cards by merchants. While Little & Co. may be able to provide some information, merchants are responsible for the awareness of and adherence to all applicable regulatory requirements imposed by the Internal Revenue Service, other government agencies, and other interested parties (for example, Visa, MasterCard, SIGIS, etc.)

- Merchants must become a member of the Special Interest Group for IIAS Standards (SIGIS)
- Merchants must obtain and maintain SIGIS certification
- Merchants, except those with healthcare related MCCs, must have an Inventory Information Approval System (IIAS) used to identify eligible healthcare purchases as defined and required by the Internal Revenue Code.
- Merchants must obtain special account numbers from Visa and MasterCard to process these transactions
- Merchants must support data retention and retrieval of line item details for eligible healthcare products included in Healthcare Card transactions

- Merchants must complete Little Certification Testing for the use of this feature, as well as the Partial Auth feature.
- Transactions must include the `IIASFlag` element set to **Y**.

1.12 Supported Transaction Types

Little Batch processing supports all transaction types except Voids and eCheck Voids, which are handled as Online transactions only. Online processing handles all transaction types. This section provides a description of each transaction type, information concerning its use, and any special considerations.

1.12.1 Authorization Transaction

The Authorization transaction enables you to confirm that a customer has submitted a valid payment method with their order and has sufficient funds to purchase the goods or services they ordered. Setting the `<allowPartialAuth>` element to **true** in the Authorization request enables the system to return authorizations for a portion of the order amount for cases where the card does not have an adequate credit limit or balance available for the full amount.

An approved Authorization reduces the customer's credit limit (or bank balance, in the case of a debit card) by the amount of the approval by placing the amount on hold. If you have the Prepaid Indicator feature enabled, the Authorization response also includes an element that indicates if the card is Prepaid, as well as an element indicating the available balance on the card.

NOTE: While most merchants perform Authorizations as Online transactions, there is no requirement to do so.

The lifespan of an Authorization depends upon the payment method. [Table 1-4](#) provides information concerning Authorization lifespans for various card types and alternate payment methods.

TABLE 1-4 Lifespan of Payment Authorizations

Payment Type	Lifespan of Authorization
MasterCard	7 days
Visa	7 days
American Express	7 days
Discover	30 days
PayPal	29 days total by default; Little & Co. recommends capture submission within three days. For more information, see the <i>Little & Co. PayPal Integration Guide</i> .
Bill Me Later	30 days by default; for more information about Bill Me Later authorizations, see the <i>Little & Co. Bill Me Later Integration Guide</i> .

As long as the authorization has not expired, or the amount exhausted, you can use it repeatedly to fulfill an order. This would be the case if the Authorization covered multiple items with staggered deliveries. In this scenario, you would issue a Partial Capture transaction as each item shipped until the order was completely fulfilled.

NOTE: If you obtain an Authorization through approved vendors for voice and terminal authorizations, you would use a Capture Given Auth transaction to deposit the funds (see [Capture Given Auth Transaction](#) on page 38).

1.12.1.1 AVS Only Transaction

An AVS Only transaction is a variation of an Authorization transaction that uses the Address Verification System to enable you to verify that a customer supplied address matches the billing address associated with the card. To submit an AVS Only transaction, submit an Authorization transaction with the `<amount>` element set to 000 and the optional `billToAddress` element with appropriate child values.

1.12.2 Authorization Reversal Transactions

The primary use of Authorization Reversal transactions is to eliminate any unused amount on an unexpired Authorization. Issuing an Authorization Reversal has the benefit of freeing any remaining held amount that reduces the buying power of your customer. Potentially, this both increases customer satisfaction and can allow them to proceed with additional purchases that may otherwise be blocked by credit limits. It also helps you avoid ant misuse of Auth fees imposed by the card associations.

NOTE: For American Express transactions, the reversal amount must match the authorization amount. Partial reversals and reversals against remaining amount after a partial capture are not allowed. Attempts to perform these types of reversals result in a Response Code of 336 - Reversal amount does not match Authorization amount.

For example, consider the following scenario. A customer with a \$6,000 credit limit orders a \$3,000 stereo system, but the stereo is a discontinued item. The merchant notifies the customer, but does not perform an Authorization Reversal. The customer attempts to submit a new order for a \$3,001 stereo.

- If the customer uses the same credit card for both orders, the second order could be denied, since the account's remaining credit limit is only \$3,000.
- If the customer had used the same debit card for both orders, the second order places the customer's bank account in an overdraft situation (assuming a starting balance of \$6,000).

Either of these situations can result in the merchant losing a sale, as well as receiving a call from an angry customer.

Another advantage of Authorization Reversal transactions occurs on Visa transactions. In order for you to qualify for the best possible interchange rates from Visa, the amount of the Capture must match the amount of the associated Authorization. In order to take advantage of this situation for you, if the Capture amount is less than the associated Authorization amount, Litle automatically performs a partial Authorization Reversal for the unused amount (also see [Capture Request](#) on page 127).

1.12.2.1 Notes on the Use of Authorization Reversal Transactions

This section provides additional information concerning the requirements of and exceptions to the use of Authorization Reversal transactions.

- Authorization Reversal transactions are supported for the following methods of payment: PayPal, MasterCard, Visa, Discover, Diners Club, and JC and American Express, but American Express and PayPal only support reversals of the entire Authorized amount (no partial reversals).
- All transactional data, including associated Authorizations, Captures, and Refunds, must be LitleXML format.
- Litle & Co. recommends that you send the Authorization Reversal transaction using the same submission method (Batch or Online) as used for the Capture transaction. This is to eliminate a possible race condition that may occur if you submit an Online Authorization Reversal prior to the processing of a Batch containing the associated Capture transaction.
- For Batch transactions, send Authorization Reversal transactions in a session separate from the both the associated Authorization and the associated Capture transactions.
- For Online transactions, when following an Authorization with an Auth Reversal, allow a minimum of one minute between the transactions.
- For Visa transactions, Litle automatically performs a partial Authorization Reversal, if the Capture amount is less than the associated Authorization amount.
- If you do not specify an amount (`<amount>` element) in the Authorization Reversal, Litle reverses the total amount of the associated Authorization.
- Do not send an Authorization Reversal against an expired Authorization. This results in a *306 - Auth expired, so amount does not need to be reversed* error. When an Authorization expires, the hold amount is automatically reversed.
- Do not send an Authorization Reversal against an Authorization that does not exist in our system. For example, if you sends a reversal against an Authorization that failed or an Authorization that was declined, the Authorization Reversal returns a *360 - No transaction found with specified litleTxnId* error.
- Do not send an Authorization Reversal against a payment type that does not support Authorization Reversals. This results in a *335 - This method of payment does not support reversals* error.

- Do not send an Authorization Reversal for a fully depleted Authorization. This results in a *111 - Authorization amount has already been depleted* error.

1.12.2.2 Using Authorization Reversal to Halt Recycling Engine

If you are using the Little Recycling Engine to optimize your authorizations and need to discontinue the automatic recycling of the transaction, you use an Authorization Reversal transaction to halt the retries. For example, if a customer cancels an order and the authorization for the order is being retried by the Recycling Engine, you submit an Authorization Reversal transaction to halt the automatic recycling of the authorization.

NOTE: If the initial transaction you submitted is a Sale (conditional deposit) rather than an Authorization, you use a Void transaction to halt the recycling.

1.12.3 Capture Transaction

You use a Capture transaction to transfer previously authorized funds from the customer to you after order fulfillment. You can submit a Capture transaction for the full amount of the Authorization, or for a lesser amount by setting the `partial` attribute to `true`.

NOTE: For a Visa transaction, if you submit a Capture for an amount less than the Authorized amount, Litle automatically issues a partial Authorization Reversal for the balance of the Authorized amount.

1.12.4 Capture Given Auth Transaction

Similar to a Capture transaction, you use a Capture Given Auth transaction to transfer previously authorized funds from the customer to you after fulfillment. However, you typically use a Capture Given Auth transaction if the associated Authorization occurred outside of the Litle & Co. system (for example, if you received a telephone Authorization). Another possible use for a Capture Given Auth transaction is if the Authorization transaction occurred within the Litle system, but the `litletxnId` is unknown by the submitting party (for example, if the Auth was submitted by a merchant, but a fulfiller submits a Capture Given Auth).

Whenever you submit a Capture Given Auth transaction, Litle attempts to match it to an existing Authorization using COMAAR data (Card Number, Order Id, Merchant Id, Amount, Approval Code, and (Auth) Response Date) in order to obtain a better Interchange rate for the transaction. The application uses the following matching logic:

- If the Order Id was either not submitted (blank, spaces, or null) or does not match any Auth in the system, it is ignored and the matching attempt proceeds using the remaining COMAAR data.
- If the matching operation results in multiple possible matches, the application selects the Authorization with the lowest amount that is greater than or equal to the Capture Given Auth amount.

NOTE: In all cases, the Authorization amount must always be greater than or equal to the Capture Given Auth amount.

- If necessary, the application further narrows the match candidates to the one with the most recent response date.

NOTE: If Litle is able to match the Capture Given Auth to an Authorization and the following conditions are met: the card type is Visa and the Capture Given Auth amount is less than the Authorization amount, then Litle will issue an Auth Reversal transaction for the balance of the Authorization.

This is done to obtain the best possible interchange rates from Visa.

1.12.5 Credit Transaction

You use a Credit transaction to refund money to a customer, even if the original transaction occurred outside of the Little & Co. system. You can submit refunds against any of the following payment transactions:

- Capture Transaction
- Capture Given Auth Transaction
- Force Capture Transaction
- Sale Transaction
- External Sale or Capture

NOTE: Little recommends that all Credit transactions in a Batch be sent separate from the associated Capture or Sale transactions.

1.12.6 eCheck Credit Transaction

Similar to a Credit transaction, you use an eCheck Credit transaction to refund money to a customer, but only when the method of payment was an eCheck. You can submit an eCheck Credit transaction regardless of whether the original transaction occurred in or out of the Little & Co. system.

1.12.7 eCheck Redeposit Transaction

You use this transaction type to manually attempt redeposits of eChecks returned for either Insufficient Funds or Uncollected Funds. You can use this element in either Online or Batch transactions.

NOTE: Do not use this transaction type if you are enabled for the Auto Redeposit feature. If you are enabled for the Auto Redeposit feature, the system will reject any `echeckRedeposit` transaction you submit.

1.12.8 eCheck Sales Transaction

You use an eCheck Sales transaction to transfer funds from the customer to you after order fulfillment. It is the eCheck equivalent of a Capture transaction. Funding usually occurs within two days. You can also submit this transaction type as a conditional capture, which makes the processing of the deposit conditional upon a successful verification. If the verification fails, the deposit is not processed.

1.12.9 eCheck Verification Transaction

You use an eCheck Verification transaction to initiate a comparison to a database containing information about checking accounts. The database may include information as to whether the account has been closed, as well as whether there is a history of undesirable behavior associated with the account/account holder.

1.12.10 eCheck Void Transaction (Online Only)

You use an eCheck Void transaction to either halt automatic redeposit attempts of eChecks returned for either Insufficient Funds or Uncollected Funds, or cancel an eCheck Sale transaction, as long as the transaction has not yet settled. This also applies to merchant initiated redeposits. You can use this element only in Online transactions.

1.12.11 Force Capture Transaction

A Force Capture transaction is a Capture transaction used when you do not have a valid Authorization for the order, but have fulfilled the order and wish to transfer funds.

CAUTION: Merchants must be authorized by Little & Co. before processing this transaction. In some instances, using a Force Capture transaction can lead to chargebacks and fines.

1.12.12 Sale Transaction

The Sale transaction enables you to both authorize fund availability and deposit those funds by means of a single transaction. The Sale transaction is also known as a conditional deposit, because the deposit takes place only if the authorization succeeds. If the authorization is declined, the deposit will not be processed.

NOTE: If the authorization succeeds, the deposit will be processed automatically, regardless of the AVS or CVV2 response.

1.12.13 Void Transaction (Online Only)

The Void transaction enables you to cancel any settlement transaction as long as the transaction has not yet settled and the original transaction occurred within the Little system (Voids require a reference to a littleTxnId).

NOTE: Do not use Void transactions to void an Authorization. To remove an Authorization use an Authorization reversal transaction (see [Authorization Reversal Transactions](#) on page 35.)

1.12.13.1 Using Void to Halt Recycling Engine

If you use the Little Recycling Engine service and Sale transactions (conditional deposits) to authorize and capture the funds, you must use a Void transaction to discontinue the automatic recycling of the transaction should the need arise. For example, if a customer cancels an order and the Sale transaction is being retried by the Recycling Engine, you submit a Void transaction to halt the automatic recycling of the transaction.

NOTE: If the initial transaction you submitted is an Authorization rather than an Sale, you use an Authorization Reversal transaction to halt the recycling.



TESTING YOUR LITLEXML TRANSACTIONS

The information provided in this chapter enables you to test and verify that your submitted transaction data conforms to the required LittleXML schema. This chapter contains the following topics:

- [Certification Environment Usage Policy](#)
- [Overview of Testing](#)
- [Transferring Files](#)
- [Performing the Required Certification Tests](#)
- [Performing the Optional Tests](#)

2.1 Certification Environment Usage Policy

The Little Certification environment is intended to provide merchants and partners with a means to test and certify initial integration code, as well as enhanced features and functionalities.

2.1.1 Limitations

The Little Certification environment is not designed or supported to be a high availability, production-level processing system, nor does it support true capacity or volume testing. While we currently allow merchants and partners the ability to maintain an open test account once they are live and processing with us, the intent of the system and the support provided is specific to a certification-level environment (i.e. the hardware and software are not production-level).

IMPORTANT: Little reserves the right to terminate a user's access to the Certification environment if it is determined that usage falls outside of the realm of certification-level testing, and whose subsequent misuse may adversely affect the overall health of the system and the ability of other Little merchants and/or partners to properly access and utilize the Certification environment.

Transactional Limits

In order to ensure that the Little Certification environment is utilized as intended, the following processing transactions limits will be imposed:

- Daily maximum of 1000 Online transactions
- Daily maximum of 10000 Batch transactions

IP Address Limits

A maximum of five (5) IP addresses will be allocated per merchant or partner for access to the Little Certification environment.

Concurrent Connection Limits (Online Processing)

A maximum of three (3) concurrent connections will be granted per IP address for each Little merchant or partner.

2.1.2 Regularly Scheduled Maintenance Window

Little & Co. will designate a recurring weekly maintenance window of Tuesdays from 8-10 AM ET for the Certification environment. The intent is that, to the best of Little's ability, any systematic updates and/or maintenance to the Certification environment will be conducted during this maintenance window.

2.1.3 Data Retention Policy

Typically, the Little & Co. Certification environment will be upgraded and re-started during the maintenance window on the third Tuesday of every month. All transactional data, including sessions, batches and payment transactions, will be purged. Merchants and partners can therefore expect from zero to thirty days of transactional data to be present in the Little Certification database.

Little & Co. reserves the right to modify the current Data Retention Policy at any time, if it is determined that the current amount of maintained historical data is negatively impacting the functionality of the Certification environment.

If you have any questions, please contact your Little Implementation Consultant or Relationship Manager.

2.2 Overview of Testing

The purpose of the testing and certification process is to verify that your order entry and supporting systems construct and send xml messages that comply with the LittleXML requirements. The Little & Co. testing process involves submitting Little supplied data for specific fields in a request, and receiving specific data back in a response. The response returned by Little allows you to verify that you parse the LittleXML Response file correctly.

Various tables in this chapter provide the data you use while testing, including card numbers, expiration dates, transaction amounts, and other values as required by the testing process. You use this data as input to your systems resulting in structured requests that conform to the required LittleXML schema.

IMPORTANT: The test data supplied does not necessarily account for all data fields/xml elements in a particular request. Where test data is not supplied, you should provide appropriate information. You should never override your own system to enter supplied data. If you are unable to enter the supplied data without overriding your system, please consult your Implementation Consultant concerning the test and how to proceed.

Typically, Little assigns an Implementation Consultant to work with you after the completion of contract negotiations. Before you begin the testing phase, your assigned Implementation Consultant establishes a test account and supplies you with instruction for accessing the account along with the username and password. You must supply the IP address from which the data originates so we can grant access.

NOTE: Until you complete all required testing, you will only have access to the test and certification environment.

The testing process involves the following steps:

2.2.1 Planning for Certification Testing

Before you begin testing, determine which transaction types you will use according to your business needs. Virtually everyone will make use of the following basic transaction types: Authorization, Capture, Sale, Credit, and Void. There are several other transaction types and most transaction types offers many options/optional fields that you may wish to utilized and therefore test. For example, if you decide to offer eChecks as a alternate payment method, there are several associated certification tested required. Similarly, if you elect to make use of the Smart Authorization feature set, there are additional Authorization tests required for certification.

NOTE: For information about certification testing for PayPal, Bill Me Later, Account Updater Service, PayPage and Chargeback XML feed, please refer to the following documents:

- *Litle & Co. PayPal Integration Guide*
 - *Litle & Co. Bill Me Later Integration Guide*
 - *Litle & Co. Account Updater Implementation and Operations Guide*
 - *Litle & Co. Pay Page Integration Guide*
 - *Litle & Co. Chargeback XML and Support Documentation API Reference Guide*
-

2.2.2 Required Certification Testing

Certification testing is a required phase of implementing the LitleXML format. During the certification testing, a Litle & Co. Implementation Consultant works with you to verify that your transaction submissions meet the requirements of the LitleXML specifications. Each transaction type has specific test scenarios that use specific data sets simulating real transactions. The Litle Certification environment responds to each submission with an XML response message, allowing you to verify that you have coded correctly to parse and store the transaction data returned to you. For more detailed information, see [Performing the Required Certification Tests](#) on page 53.

2.2.3 Optional Testing

Litle & Co. provides you with test data, test scenarios, a test environment, and credentials for using that test environment so you can perform these tests on your own keeping in mind the limitations of the certification environment (see [Certification Environment Usage Policy](#) on page 44). During unattended testing, you use these resources to perform all of the tests that apply to your business needs.

NOTE: If you have questions or need assistance while performing unattended testing, feel free to call your assigned Implementation Consultant or email implementation@litle.com.

2.3 Transferring Files

As discussed in [Communications Protocols](#) on page 3, there are several protocols you can use to submit your transactions to Litle & Co. for processing. This section provides additional information concerning the recommended methods for transferring your LitleXML Batch and Online transaction files.

2.3.1 Transferring Batch Files

While you can submit your Batch files via HTTPS POST, Litle recommends batch submission using either FTP or sFTP. This section describes how to FTP your files (not test system specific) and includes the following topics:

- [Submitting a Batch File for Processing](#)
- [Retrieving Processed Batch Files](#)

NOTE: Before you begin transferring files via FTP, Litle & Co. provides the FTP Host and a username/password for the Litle & Co. test system.

2.3.1.1 Submitting a Batch File for Processing

CAUTION: File naming conventions are crucial to the file submission process. Incorrect file names will prevent the file from being processed or may stop processing due to an incomplete file transfer.

Do not append .asc to the end of the filename (Step 3). You must replace the .prg extension with .asc. If .prg appears in the filename, the system will not process the file

1. On your local system, add the extension .prg (lowercase) to the name of the file you want to submit. For example, you could name the file MerchantName_YYMMDD.prg. Keep in mind the following rules:
 - Spaces are not allowed in the file name
 - The .prg extension must be lower case
2. Open your FTP connection to the Litle & Co. inbound directory and move your file to the Litle & Co. directory.
3. After the ftp process completes, change the extension of the transmitted file (in the Litle & Co. inbound directory) from .prg to .asc (lowercase). The system polls the directory for files with an .asc extension every thirty seconds. When the system encounters files with the proper extension, it retrieves them for processing.

2.3.1.2 Retrieving Processed Batch Files

Depending on the size of your file, your response should be ready within a few minutes. Batch files containing large number of transactions take longer. For example, a batch of 10,000 transactions may require as long as ten minutes to process.

The initial response represents an acknowledgement that we received the transactions and notification that we will deliver them upstream to Visa and/or MasterCard for review. Since we perform validation operations against the credit card number and the expiration date, you may also receive a decline responses containing the appropriate response code.

To retrieve response files from the outbound directory:

NOTE: Little & Co. removes response files from the outbound directory after 24 hours. Plan to retrieve your files daily.

1. Open your FTP connection to the Little & Co. outbound directory.
2. Locate the response file, which will have the same name as the file you submitted. If the response file has a .prg extension, it is still transferring. The extension changes to .asc when the transfer to the outbound directory completes.
3. Retrieve the response file.

2.3.2 Transferring Online Files

The recommended method for submitting Online transactions is via HTTPS POST. The sections that follow provide examples of ASP and Java programming methods for submitting your data using HTTPS POST.

- [ASP Programming Example](#)
- [Java Programming Example](#)
- [Helpful Web Sites](#)

NOTE: Before you begin testing, Little & Co. provides the test system URL, a username/password, and any additional information required to test your XML transactions.

2.3.2.1 ASP Programming Example

The following code is an example of a LittleXML Authorization transaction submitted via HTTPS Post using ASP.

```
Dim xml

Dim strXML

strXML = strXML & "<litleOnlineRequest version=""8.8""  
xmlns=""http://www.little.com/schema"" merchantId=""MERCHANTID"">"

strXML = strXML & "<authentication>"

strXML = strXML & "<user>USERNAME</user>"

strXML = strXML & "<password>#####</password>"

strXML = strXML & "</authentication>"

strXML = strXML & "<authorization id=""834262""  
reportGroup=""123"" customerId=""038945"">"

strXML = strXML & "<orderId>3235059</orderId>"

strXML = strXML & "<amount>54399</amount>"

strXML = strXML & "<orderSource>ecommerce</orderSource>"

strXML = strXML & "<billToAddress>"

strXML = strXML & "<name>Todd Wilson</name>"

strXML = strXML & "<addressLine1>123 Blue  
Street</addressLine1>"

strXML = strXML & "<addressLine2>Suite 108</addressLine2>"

strXML = strXML & "<addressLine3>Address Line 3</addressLine3>"

strXML = strXML & "<city>Lowell</city>"

strXML = strXML & "<state>MA</state>"

strXML = strXML & "<zip>01851</zip>"

strXML = strXML & "<country>USA</country>"

strXML = strXML & "<email>twilson@email.com</email>"

strXML = strXML & "<phone>323-222-2222</phone>"

strXML = strXML & "</billToAddress>"

strXML = strXML & "<card>"

strXML = strXML & "<type>VI</type>"

strXML = strXML & "<number>#####</number>"

strXML = strXML & "<expDate>0514</expDate>"

strXML = strXML & "<cardValidationNum>###</cardValidationNum>"

strXML = strXML & "</card>"
```

```
strXML = strXML & "</authorization>"
strXML = strXML & "</littleOnlineRequest>"
set xml = CreateObject("Microsoft.XMLHTTP")
xml.setRequestHeader "Content-type", "text/html; charset=UTF-8"
xml.Open "POST", "https://site.info.com/from_Little", False
xml.Send strXML
Response.write (xml.responseText)
set xml = Nothing
```

2.3.2.2 Java Programming Example

The following is an example of Java code used for HTTPS Post.

PostMethod and HttpClient are both part of the Apache HttpClient library located at <http://jakarta.apache.org/commons/httpclient/>.

```
PostMethod post = new PostMethod(url); // url = fully qualified url
of the server to which you are posting
post.setRequestHeader("Content-type", "text/html; charset=UTF-8");
post.setRequestBody(data); //data = request data in XML format
```

```
HttpClient client = new HttpClient();
client.setTimeout(10000); //10 second timeout (in milliseconds) is
suggested minimum, 30 second recommended for alternate payment
methods
client.executeMethod(post);
```

```
String response = post.getResponseBodyAsString();
```

```
//if the server throws an exception you get a null response
//to get around this set it to ""
if (response == null) {
    response = "";
}
post.releaseConnection();
```

2.3.2.3 Notes on Timeout Settings

While Litle & Co. optimizes our systems to ensure the return of Authorization responses as quickly as possible, some portions of the process are beyond our control. The round-trip time of an Authorization can be broken down into three parts, as follows:

1. Transmission time (across the internet) to Litle & Co. and back to the merchant
2. Processing time by the card association or authorization provider
3. Processing time Litle

Under normal operating circumstances, the transmission time to and from Litle does not exceed 0.6 seconds and processing time by Litle occurs in 0.1 seconds. Typically, the processing time by the card association or authorization provider can take between 0.5 and 3 seconds, but some percentage of transactions may take significantly longer.

Because the total processing time can vary due to a number of factors, Litle & Co. recommends using a timeout setting of 10 seconds minimum for card transactions and 30 seconds minimum for alternate payment methods. These settings should ensure that you do not frequently disconnect prior to receiving a valid authorization causing dropped orders and/or re-auths and duplicate auths.

NOTE: While it is uncommon, under certain circumstances network latency may cause a duplicate Online Sale transaction to go undetected as a duplicate. This can occur if you submit a second, duplicate Sale transaction while the response from the network for the Authorization portion of the first transaction is sufficiently delayed such that the first Sale has not been recorded as a valid transaction in the Litle system.

If you elect to submit Online Sale transactions, Litle recommends a timeout setting of not less than 60 seconds to reduce the chances of undetected duplicate Sale transactions.

2.3.2.4 Helpful Web Sites

The following web sites provide additional information, helpful hints, and examples of different programming methods used in combination with HTTPS POST.

- http://p2p.wrox.com/topic.asp?TOPIC_ID=6193
- <http://en.allexperts.com/q/Active-Server-Pages-1452/XML-ASP-Parser-2.htm>
- <http://www.java-samples.com/java/POST-toHTTPS-url-free-java-sample-program.htm>

2.4 Performing the Required Certification Tests

You are required to complete a number of certification tests prior to submitting real transaction to the Litle & Co. production system. This testing process allows you to verify that your systems not only submits transaction data correctly formatted, but also correctly parses the data returned to you in the response messages. To facilitate the certification process, Litle has established a certification environment that simulates the production environment.

During certification testing, a Litle & Co. Implementation Consultant will guide you through each required test scenario. For each transaction type specific data is supplied that you must use in your LitleXML transactions. Use of this data allows the validation of your transaction structure/syntax, as well as the return of a response file containing known data.

IMPORTANT: The test data supplied does not account for all data fields/xml elements in a particular request. Where data is not supplied, you should provide appropriate information. You should never override your own system to enter supplied data. If you are unable to enter the supplied data without overriding your system, please consult your Implementation Consultant concerning the test and how to proceed.

2.4.1 Testing Authorization, AVS Only, Capture, Credit, Sale, and Void Transactions

Table 2-1 provides 26 data sets you use to test your construction of Authorization, AVS Only, Sale and Force Capture transactions, as well as your ability to parse the information contained in the XML response messages. You also use some of the approved authorizations as the basis for testing Capture and Credit transactions.

You do not necessarily have to perform all Authorization test. The tests you perform depend upon the Litle features you have elected to use. The tests are divided as follows:

- Order Ids 1 through 9 - used to test standard Authorization requests and responses. Also used for AVS Only test and Sale test. (Capture test use the `litleTxnId` returned in the response messages.)
- Order Ids 10 through 13 - include if you plan to use partial Authorization
- Order Ids 14 and 15 - include if you plan to use the Prepaid Indicator feature (see [Prepaid Indicator](#) on page 11)
- Order Ids 16 through 19 - include if you plan to use the Affluence Indicator feature (see [Affluence Indicator](#) on page 12)
- Order Id 20 - include if you plan to use the Issuer Country feature (see [Issuer Country Indicator](#) on page 12)
- Order Ids 21 through 26 - include if you plan to use the Healthcare Card feature (see [Healthcare Card Feature](#) on page 31)

To test **Authorization** transactions:

1. Verify that your Authorization XML template is coded correctly (refer to [Authorization Transactions](#) on page 114.)
2. Submit `authorization` transactions using the data shown in the Supplied Data Elements of Order Ids 1 through 9 of [Table 2-1](#).
3. Verify that your response values match those shown in Key Response Elements for Order Ids 1 through 9 as shown in [Table 2-1](#).
4. If you wish to test **AVS only** transactions, re-submit Order Ids 1 through 5, 7, 8, and 9 (skip order 6), but substitute 000 for the amount. The AVS result returned will be the value shown in the Key Response Elements section.
5. If you plan to use **Partial Authorizations**, submit `authorization` transactions using the data shown in the Supplied Data Elements of Order Ids 10 through 13 of [Table 2-1](#).
6. Verify that your response values match those shown in Key Response Elements for Order Ids 10 through 13 as shown in [Table 2-1](#).
7. If you elected to receive **Prepaid Indicators**, submit `authorization` transactions using the data shown in the Supplied Data Elements of Order Ids 14 through 20 of [Table 2-1](#). Verify that your response values match those shown in Key Response Elements for Order Ids 14 and 15 as shown in [Table 2-1](#).
8. If you elected to receive **Affluence Indicators**, submit `authorization` transactions using the data shown in the Supplied Data Elements of Order Ids 21 through 24 of [Table 2-1](#). Verify that your response values match those shown in Key Response Elements for Order Ids 16 through 19 as shown in [Table 2-1](#).
9. If you elected to receive **Issuer Country** information, submit an `authorization` transaction using the data shown in the Supplied Data Elements of Order Id 27 of [Table 2-1](#). Verify that your response values match those shown in Key Response Elements for Order Id 27 as shown in [Table 2-1](#).
10. If you plan to handle transactions using **Healthcare (IIAS)** cards, submit `authorization` transactions using the data shown in the Supplied Data Elements of Order Ids 26 through 31 of [Table 2-1](#). Verify that your response values match those shown in Key Response Elements for Order Ids 26 through 31 as shown in [Table 2-1](#).

NOTE: Some Issuers do not return an Auth Code for \$0 Authorizations. You should code your systems to handle this possibility.

To Test **Sale** transactions:

1. Verify that your Sale XML template is coded correctly (refer to [Sale Transactions](#) on page 179.)
2. Submit `sale` transactions using the data shown in the Supplied Data Elements of Order Ids 1 through 9 of [Table 2-1](#).
3. Verify that your response values match those shown in Key Response Elements for Order Ids 1 through 9 as shown in [Table 2-1](#).

To Test **Capture** transactions:

1. Verify that your Capture XML template is coded correctly (refer to [Capture Transactions](#) on page 127.).
2. Submit `capture` transactions for Order Ids 1A through 5A using the `littleTnxId` value returned in the response messages for Authorization Order Ids 1 through 5.
3. Verify that your response values match those shown in Key Response Elements for Order Ids 1A through 5A as shown in [Table 2-1](#).

To test **Credit** transactions:

1. Verify that your Credit XML template is coded correctly (refer to [Credit Transactions](#) on page 142.)
2. Submit `credit` transactions for Order Ids 1B through 5B using the `littleTnxId` value returned in the response messages for Capture Order Ids 1A through 5A.
3. Verify that your response values match those shown in Key Response Elements for Order Ids 1B through 5B as shown in [Table 2-1](#).

To test **Void** transactions (in this test you have the option of voiding either Credit or Sale transactions):

1. Verify that your Void XML template is coded correctly (refer to [Void Transactions \(Online Only\)](#) on page 186.)
2. Submit `void` transactions for Order Ids 1C through 5C (and 6B if voiding Sale transactions) using the `littleTnxId` value returned in either the response messages for Credit Order Ids 1B through 5B, or the response messages for Sale (not Auth) Order Id 1 through 6.
3. Verify that your response values match those shown in Key Response Elements for Order Ids 1C through 5C (include 6B only if you elect to void the Sales transactions) as shown in [Table 2-1](#).

TABLE 2-1 Authorization Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
1	Authorization/Sale/AVS: <amount> 10010 <name> John Smith <addressLine1> 1 Main St. <city> Burlington <state> MA <zip> 01803-3747 <country> US <type> VI <number> 4457010000000009 <expDate> 0112 <cardValidationNum> 349		Authorization Response: <response> 000 <message> Approved <authCode> 11111 <avsResult> 01 <cardValidationResult> M	
1A	Capture: <littleTxnId>	Value returned in Auth response for Order Id 1	Capture Response: <response> 000 <message> Approved	
1B	Credit: <littleTxnId>	Value returned in Capture response for Order Id 1A	Credit Response: <response> 000 <message> Approved	
1C	Void: <littleTxnId>	Use either the value returned in the Credit response for Order Id 1B, or the value returned in the Sale response (not Auth) for Order Id 1.	Void Response: <response> 000 <message> Approved	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
2	Authorization/Sale/AVS: <amount> 20020 <name> Mike J. Hammer <addressLine1> 2 Main St. <addressLine2> Apt. 222 <city> Riverside <state> RI <zip> 02915 <country> US <type> MC <number> 5112010000000003 <expDate> 0212 <cardValidationNum> 261 <authenticationValue> BwABBJQ1AgAAA AAgJDUCAAAAAA A=		Authorization Response: <response> Approved <message> 22222 <authCode> 10 <avsResult> M <cardValidationResult> Note: Not returned <authenticationResult> for MasterCard	
2A	Capture: <littleTxnId>	Value returned in Auth response for Order Id 2	Capture Response: <response> <message>	000 Approved
2B	Credit: <littleTxnId>	Value returned in Capture response for Order Id 2A	Credit Response: <response> <message>	000 Approved
2C	Void: <littleTxnId>	Use either the value returned in the Credit response for Order Id 2B, or the value returned in the Sale response (not Auth) for Order Id 2.	Void Response: <response> <message>	000 Approved

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
3	Authorization/Sale/AVS: <amount> 30030 <name> Eileen Jones <addressLine1> 3 Main St. <city> Bloomfield <state> CT <zip> 06002 <country> US <type> DI <number> 6011010000000003 <expDate> 0312 <cardValidationNum> 758		Authorization Response: <response> Approved <message> 33333 <authCode> 10 <avsResult> M <cardValidationResult>	000
3A	Capture: <littleTxnId>	Value returned in Auth response for Order Id 3	Capture Response: <response> <message>	000 Approved
3B	Credit: <littleTxnId>	Value returned in Capture response for Order Id 3A	Credit Response: <response> <message>	000 Approved
3C	Void: <littleTxnId>	Use either the value returned in the Credit response for Order Id 3B, or the value returned in the Sale response (not Auth) for Order Id 3.	Void Response: <response> <message>	000 Approved

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
4	Authorization/Sale/AVS: <amount> 40040 <name> Bob Black <addressLine1> 4 Main St. <city> Laurel <state> MD <zip> 20708 <country> US <type> AX <number> 375001000000005 <expDate> 0412		Authorization Response: <response> Approved <message> 44444 <authCode> 12 <avsResult> blank <cardValidationResult>	000
4A	Capture: <littleTxnId>	Value returned in Auth response for Order Id 4	Capture Response: <response> <message>	000 Approved
4B	Credit: <littleTxnId>	Value returned in Capture response for Order Id 4A	Credit Response: <response> <message>	000 Approved
4C	Void: <littleTxnId>	Use either the value returned in the Credit response for Order Id 4B, or the value returned in the Sale response (not Auth) for Order Id 4.	Void Response: <response> <message>	000 Approved

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
5	Authorization/Sale/AVS: <amount> 50050 <type> VI <number> 4457010200000007 <expDate> 0512 <cardValidationNum> 463 <authenticationValue> BwABBJQ1AgAAA AAgJDUCAAAAAA A=		Authorization Response: <response> Approved <message> 55555 <authCode> 32 <avsResult> N <cardValidationResult>	000
5A	Capture: <littleTxnId>	Value returned in Auth response for Order Id 5	Capture Response: <response> <message>	000 Approved
5B	Credit: <littleTxnId>	Value returned in Capture response for Order Id 5A	Credit Response: <response> <message>	000 Approved
5C	Void: <littleTxnId>	Use either the value returned in the Credit response for Order Id 5B, or the value returned in the Sale response (not Auth) for Order Id 5.	Void Response: <response> <message>	000 Approved

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
6	Authorization/Sale: <amount> 60060 <name> Joe Green <addressLine1> 6 Main St. <city> Derry <state> NH <zip> 03038 <country> US <type> VI <number> 4457010100000008 <expDate> 0612 <cardValidationNum> 992		Authorization Response: <response> 110 <message> Insufficient Funds blank <authCode> 34 <avsResult> P <cardValidationResult>	
6A	Void: <littleTxnId> Use the value returned in the Sale response (not Auth) for Order Id 6.		Void Response: <response> 360 <message> No transaction found with specified littleTxnId	
7	Authorization/Sale/AVS: <amount> 70070 <name> Jane Murray <addressLine1> 7 Main St. <city> Amesbury <state> MA <zip> 01913 <country> US <type> MC <number> 5112010100000002 <expDate> 0712 <cardValidationNum> 251		Authorization Response: <response> 301 Invalid Account Number <message> blank <authCode> 34 <avsResult> N <cardValidationResult>	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
8	Authorization/Sale/AVS: <amount> 80080 <name> Mark Johnson <addressLine1> 8 Main St. <city> Manchester <state> NH <zip> 03101 <country> US <type> DI <number> 6011010100000002 <expDate> 0812 <cardValidationNum> 184		Authorization Response: <response> Call Discover <message> blank <authCode> 34 <avsResult> P <cardValidationResult>	123
9	Authorization/Sale/AVS: <amount> 90090 <name> James Miller <addressLine1> 9 Main St. <city> Boston <state> MA <zip> 02134 <country> US <type> AX <number> 3750010100000003 <expDate> 0912 <cardValidationNum> 0421		Authorization Response: <response> Pick Up Card <message> blank <authCode> 34 <avsResult> blank <cardValidationResult>	303
NOTE: The data sets for orderId 10 through 13 are designed to test Authorization transactions resulting in partial authorizations. If you are not coding to use partial authorizations, you may skip these tests.				
10	<amount> 40000 <type> VI <number> 4457010140000141 <expDate> 0912 <allowPartialAuth> true		<response> 010 <message> Partially Approved <approvedAmount> 32000	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
11	<amount> 60000 <type> MC <number> 5112010140000004 <expDate> 1111 <allowPartialAuth> true		<response> 010 <message> Partially Approved <approvedAmount> 48000	
12	<amount> 50000 <type> AX <number> 3750010140000009 <expDate> 0412 <allowPartialAuth> true		<response> 010 <message> Partially Approved <approvedAmount> 40000	
13	<amount> 15000 <type> DI <number> 6011010140000004 <expDate> 0812 <allowPartialAuth> true		<response> 010 <message> Partially Approved <approvedAmount> 12000	
NOTE: The data sets for orderId 14 through 20 are designed to test Authorization transactions that return Prepaid Indicator information in the response message. If you are not coding to use the optional Prepaid Indicator feature of the Smart Authorization feature set, you may skip these tests.				
14	<amount> 3000 <type> VI <number> 4457010200000247 <expDate> 0812		<response> 000 <message> Approved <type> PREPAID <availableBalance> 2000 <reloadable> NO <prepaidCardType> GIFT	
15	<amount> 3000 <type> MC <number> 5500000254444445 <expDate> 0312		<response> 000 <message> Approved <type> PREPAID <availableBalance> 2000 <reloadable> YES <prepaidCardType> PAYROLL	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
16	<amount> 3000 <type> MC <number> 5592106621450897 <expDate> 0312		<response> 000 <message> Approved <type> PREPAID <availableBalance> 0 <reloadable> YES <prepaidCardType> PAYROLL	
17	<amount> 3000 <type> MC <number> 5590409551104142 <expDate> 0312		<response> 000 <message> Approved <type> PREPAID <availableBalance> 6500 <reloadable> YES <prepaidCardType> PAYROLL	
18	<amount> 3000 <type> MC <number> 5587755665222179 <expDate> 0312		<response> 000 <message> Approved <type> PREPAID <availableBalance> 12200 <reloadable> YES <prepaidCardType> PAYROLL	
19	<amount> 3000 <type> MC <number> 5445840176552850 <expDate> 0312		<response> 000 <message> Approved <type> PREPAID <availableBalance> 20000 <reloadable> YES <prepaidCardType> PAYROLL	
20	<amount> 3000 <type> MC <number> 5390016478904678 <expDate> 0312		<response> 000 <message> Approved <type> PREPAID <availableBalance> 10050 <reloadable> YES <prepaidCardType> PAYROLL	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
NOTE: The data sets for orderId 21 through 24 are designed to test Authorization transactions that return Affluence Indicator information in the response message. If you are not coding to use the optional Affluence Indicator feature of the Smart Authorization feature set, you may skip these tests.				
21	<amount> <type> <number> <expDate>	5000 VI 4457010201000246 0912	<response> <message> <affluence>	000 Approved AFFLUENT
22	<amount> <type> <number> <expDate>	5000 VI 4457010202000245 1111	<response> <message> <affluence>	000 Approved MASS AFFLUENT
23	<amount> <type> <number> <expDate>	5000 MC 5112010201000109 0412	<response> <message> <affluence>	000 Approved AFFLUENT
24	<amount> <type> <number> <expDate>	5000 MC 5112010202000108 0812	<response> <message> <affluence>	000 Approved MASS AFFLUENT
NOTE: The data set for orderId 25 is designed to test Authorization transactions that return Issuer Country information in the response message. If you are not coding to use the optional Issuer Country feature of the Smart Authorization feature set, you may skip these tests.				
25	<amount> <type> <number> <expDate>	5000 VI 4100204446270000 1112	<response> <message> <issuerCountry>	000 Approved BRA

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
NOTE: The data sets for orderId 26 through 31 are designed to test Authorization transactions involving Healthcare Care (IIAS) transaction. If you are not coding to use the optional Healthcard Card feature of the Smart Authorization feature set, you may skip these tests.				
26	<amount> <type> <number> <expDate> <allowPartialAuth> <totalHealthcareAmount>	18698 MC 5194560012341234 1212 true 20000	<response> <message>	341 Invalid healthcare amounts
27	<amount> <type> <number> <expDate> <allowPartialAuth> <totalHealthcareAmount> <RxAmount>	18698 MC 5194560012341234 1212 true 15000 16000	<response> <message>	341 Invalid healthcare amounts
28	<amount> <type> <number> <expDate> <allowPartialAuth> <totalHealthcareAmount> <RxAmount>	15000 MC 5194560012341234 1212 true 15000 3698	<response> <message>	000 Approved
29	<amount> <type> <number> <expDate> <allowPartialAuth> <totalHealthcareAmount> <RxAmount> <visionAmount> <clinicOtherAmount> <dentalAmount>	18699 VI 4024720001231239 1212 true 31000 1000 19901 9050 1049	<response> <message>	314 Invalid healthcare amounts

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
30	<amount> 20000 <type> VI <number> 4024720001231239 <expDate> 1212 <allowPartialAuth> true <totalHealthcareAmount> 20000 <RxAmount> 1000 <visionAmount> 19901 <clinicOtherAmount> 9050 <dentalAmount> 1049		<response> 341 <message> Invalid healthcare amounts	
31	<amount> 25000 <type> VI <number> 4024720001231239 <expDate> 1212 <allowPartialAuth> true <totalHealthcareAmount> 18699 <RxAmount> 1000 <visionAmount> 15099		<response> 010 <message> Partially Approved <approvedAmount> 18699	

2.4.1.1 Testing Recycling Advice

If you have elected to receive Authorization Recycling Advice, you must complete the certification tests in this section. The primary purpose of these tests are for you to verify that your systems correctly parse the recycle advice returned in the response message and that your systems can act on the recommendations.

If you are not coding to receive Recycling Advice, skip this test and go to [Testing Authorization Reversal Transactions](#) on page 69.

To test the Recycling Advice feature:

1. Submit `authorization` transactions using the orders shown in [Table 2-2](#).
2. The response messages for each transaction will contain a `recycling` element (see [recycling](#) on page 378). Verify that your system parses the advice correctly and sets-up to recycle the Auth according to the advice provided in the Key Response Elements for the given `orderId`.
3. Recycle the Auth according to the advice provided.
 - For Order Id **VIcredit12401** the response message for the second recycle attempt (third Auth transaction) will contain the `recycleAdviceEnd` element.
 - For Order Id **VIcredit13201** the response message for the second recycle attempt (third Auth transaction) will contain additional recycling advice, but you can discontinue the testing at this point.

TABLE 2-2 Authorization Recycling Advice Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
VIcredit12401	<amount>	12401	Initial Auth Response:	
	<type>	VI	<response>	110
	<number>	4457012400000001	<message>	Insufficient Funds
	<expDate>	1220	<nextRecycleTime>	<i>Auth Date + 1 Day</i>
			First Recycle Attempt:	
			<response>	110
			<message>	Insufficient Funds
			<nextRecycleTime>	<i>(Original) Auth Date + 2 Days</i>
			Second Recycle Attempt:	
			<response>	110
			<message>	Insufficient Funds
			<recycleAdviceEnd>	End of Advice

TABLE 2-2 Authorization Recycling Advice Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
Vlprepaid13201	<amount>	13201	Initial Auth Response:	
	<type>	VI		
	<number>	4457013200000001		
	<expDate>	1220		
			<response>	349
			<message>	Do Not Honor
			<nextRecycleTime>	<i>Auth Date + 1 Day</i>
			First Recycle Attempt:	
			<response>	349
			<message>	Do Not Honor
			<nextRecycleTime>	<i>(Original) Auth Date + 3 Days</i>
			Second Recycle Attempt:	
			<response>	349
			<message>	Do Not Honor
			<recycleAdviceEnd>	<i>(Original) Auth Date + 5 Days</i>

2.4.2 Testing Authorization Reversal Transactions

If you plan to use Authorization Reversal Transactions, you must perform this test. If you do not plan to use Authorization Reversal transactions, skip this test and go to [Testing eCheck Transactions](#) on page 73.

To test Authorization Reversal Transactions:

1. Verify that your Authorization Reversal XML templates are coded correctly (refer to [Authorization Reversal Transactions](#) on page 124).
2. Submit the Authorizations, Captures (if applicable), and Authorization Reversal Transactions using the orders shown in [Table 2-3](#).
3. Verify that your response values match those shown in Key Response Elements as shown in [Table 2-3](#).

TABLE 2-3 Authorization Reversal Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
32	Authorization: <amount> 10010 <name> John Smith <addressLine1> 1 Main St. <city> Burlington <state> MA <zip> 01803-3747 <country> US <type> VI <number> 4457010000000009 <expDate> 0112 <cardValidationNum> 349		Authorization Response: <response> 000 <message> Approved <authCode> 11111 <avsResult> 01 <cardValidationResult> M	
32A	Capture: <amount> 5005 <littleTxnId> Value returned in Auth response for Order Id 27		Capture Response: <response> 000 <message> Approved	
32B	Authorization Reversal: <littleTxnId> Value returned in Auth response for Order Id 27 <amount> blank		Auth Reversal Response: <response> 111 <message> Authorization amount has already been depleted Note: This transaction returns 111 instead of 000, because it is unnecessary to submit an Authorization Reversal for the Visa payment card.	

TABLE 2-3 Authorization Reversal Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
33	Authorization: <amount> 20020 <name> Mike J. Hammer <addressLine1> 2 Main St. <addressLine2> Apt. 222 <city> Riverside <state> RI <zip> 02915 <country> US <type> MC <number> 5112010000000003 <expDate> 0212 <cardValidationNum> 261 <authenticationValue> BwABBJQ1AgAAA AAgJDUCAAAAAA A=		Authorization Response: <response> Approved <message> 22222 <authCode> 10 <avsResult> M <cardValidationResult> <authenticationResult>	000 Note: Not returned for MasterCard
33A	Authorization Reversal: <littleTxnId> <amount>	Value returned in Auth response for Order Id 28 blank	Auth Reversal Response: <response> <message>	000 Approved
34	Authorization: <amount> 30030 <name> Eileen Jones <addressLine1> 3 Main St. <city> Bloomfield <state> CT <zip> 06002 <country> US <type> DI <number> 6011010000000003 <expDate> 0312 <cardValidationNum> 758		Authorization Response: <response> Approved <message> 33333 <authCode> 10 <avsResult> M <cardValidationResult>	000

TABLE 2-3 Authorization Reversal Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
34A	Authorization Reversal: <littleTxnId> <amount>	Value returned in Auth response for Order Id 29 blank	Auth Reversal Response: <response> <message>	000 Approved
35	Authorization: <amount> <name> <addressLine1> <city> <state> <zip> <country> <type> <number> <expDate>	40040 Bob Black 4 Main St. Laurel MD 20708 US AX 375001000000005 0412	Authorization Response: <response> <message> <authCode> <avsResult> <cardValidationResult>	000 Approved 44444 12 blank
35A	Capture: <amount> <littleTxnId>	20020 Value returned in Auth response for Order Id 30	Capture Response: <response> <message>	000 Approved
35B	Authorization Reversal: <littleTxnId> <amount>	Value returned in Auth response for Order Id 30 20020	Auth Reversal Response: <response> <message>	000 Approved
36	Authorization: <amount> <type> <number> <expDate>	20500 AX 375000026600004 0512	Authorization Response: <response> <message>	000 Approved

TABLE 2-3 Authorization Reversal Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
36A	Authorization Reversal: <littleTxnId> <amount>	Value returned in Auth response for Order Id 31 10000	Auth Reversal Response: <response> <message>	336 Reversal Amount does not match Authorization amount

2.4.3 Testing eCheck Transactions

If you have elected to offer eChecks as an alternate payment method, you are required to complete the tests in this section. Data sets are provided for you to use to test your construction of XML request messages, as well as the parsing of the response messages for eCheck transactions.

NOTE: The eCheck Verification test is required only if you plan to perform eCheck Verifications.

To test **eCheck Verification** transactions:

1. Verify that your eCheck XML template is coded correctly (refer to [eCheck Verification Transactions](#) on page 162.)
2. Submit the eCheck Verification transactions using the data sets supplied in [Table 2-4](#).

NOTE: In addition to the test data provided in the table, you must also provide appropriate data for other child elements of the `billToAddress` element, such as `Address1` (2, 3), `city`, `state`, etc.

For Corporate accounts (Order ID 39 and 40) you must include the `firstName`, `lastName`, and `companyName` in the `echeckVerification` request.

3. Verify that your response values match those shown in the Key Response Elements section of [Table 2-4](#).
4. After you complete this test, go to the next eCheck test.

TABLE 2-4 eCheck Verification Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
37	<amount> <orderSource> <firstName> <lastName> <accType> <accNum> <routingNum>	3001 telephone Tom Black Checking 10@BC99999 053100300	<response> <message>	301 Invalid Account Number
38	<amount> <orderSource> <firstName> <lastName> <accType> <accNum> <routingNum>	3002 telephone John Smith Checking 1099999999 053000219	<response> <message>	000 Approved
39	<amount> <orderSource> <firstName> <lastName> <companyName> <accType> <accNum> <routingNum>	3003 telephone Robert Jones Good Goods Inc Corporate 3099999999 053100300	<response> <message>	950 Declined - Negative Information on File
40	<amount> <orderSource> <firstName> <lastName> <companyName> <accType> <accNum> <routingNum>	3004 telephone Peter Green Green Co Corporate 8099999999 063102152	<response> <message>	951 Absolute Decline

To test **eCheck Sale** transactions:

1. Verify that your eCheck XML template is coded correctly (refer to [eCheck Sale Transactions](#) on page 159).
2. Submit the `echeckSale` transactions using the Supplied Data elements shown in [Table 2-5](#).

NOTE: In addition to the test data provided in the table, you must also provide appropriate data for other child elements of the `billToAddress` element, such as `Address1` (2, 3), `city`, `state`, etc.

3. Verify that your response values match those shown in [Table 2-5](#).
4. After you complete this test, go to the next test.

TABLE 2-5 eCheck Sale Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
41	<code><amount></code> <code><orderSource></code> <code><firstName></code> <code><middleInitial></code> <code><lastName></code> <code><accType></code> <code><accNum></code> <code><routingNum></code>	2008 telephone Mike J Hammer Checking 10@BC99999 053100300	<code><response></code> <code><message></code>	301 Invalid Account Number
42	<code><amount></code> <code><orderSource></code> <code><firstName></code> <code><lastName></code> <code><accType></code> <code><accNum></code> <code><routingNum></code>	2004 telephone Tom Black Checking 4099999992 211370545	<code><response></code> <code><message></code>	000 Approved

TABLE 2-5 eCheck Sale Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
43	<amount> 2007 <orderSource> telephone <firstName> Peter <lastName> Green <companyName> Green Co <accType> Corporate <accNum> 6099999992 <routingNum> 211370545		<response> 000 <message> Approved Note: This response will include the accountUpdater element (see accountUpdater on page 195).	
44	<amount> 2009 <orderSource> telephone <firstName> Peter <lastName> Green <companyName> Green Co <accType> Corporate <accNum> 9099999992 <routingNum> 053133052		<response> 900 <message> Invalid Bank Routing Number	

To test **eCheck Credit** transactions:

1. Verify that your eCheck XML template is coded correctly (refer to [eCheck Credit Transactions](#) on page 152.)

NOTE: In addition to the test data provided in the table, you must also provide appropriate data for other child elements of the `billToAddress` element, such as `Address1` (2, 3), `city`, `state`, etc.

2. Submit the `echeckCredit` transactions using the data in the Supplied Data Elements section of [Table 2-6](#).
3. Verify that your response values match those shown in the Key Response Elements section of [Table 2-6](#).
4. After you complete this test, go to the next test.

TABLE 2-6 eCheck Credit Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
45	<amount> <orderSource> <firstName> <lastName> <accType> <accNum> <routingNum>	1001 telephone John Smith Checking 10@BC99999 053100300	<response> <message>	301 Invalid Account Number
46	<amount> <orderSource> <firstName> <lastName> <companyName> <accType> <accNum> <routingNum>	1003 telephone Robert Jones Widget Inc Corporate 3099999999 063102152	<response> <message>	000 Approved
47	<amount> <orderSource> <firstName> <lastName> <companyName> <accType> <accNum> <routingNum>	1007 telephone Peter Green Green Co Corporate 6099999993 211370545	<response> <message>	000 Approved Note: This response will include the accountUpdater element (see accountUpdater on page 195.)
48	<littleTxnId>	Value returned in eCheck Sale response for Order Id 36	<response> <message>	000 Approved
49	<littleTxnId>	2	<response> <message>	360 No transaction found with specified littleTxnId

To test **eCheck Void** transactions:

1. Verify that your eCheck XML template is coded correctly (refer to [eCheck Void Transactions \(Online Only\)](#) on page 166.)
2. (Re)Submit the `echeckSale` transaction for Order ID #42 as shown in [Table 2-5](#) with a different value for the `id` attribute.
 - a. Using the `littleTxnId` returned in the `echeckSaleResponse` message, submit an `echeckVoid` transaction.
 - b. The system returns an `echeckVoidResponse` Response Code of “000” and a message of “**Approved**”.
3. Using the `littleTxnId` returned in the `echeckCreditResponse` message for Order ID #41, submit an `echeckVoid` transaction.
 - a. The system returns an `echeckVoidResponse` Response Code of “000” and a message of “**Approved**”.
4. Submit an `echeckVoid` request using a value of “2” for the `littleTxnId`.
 - a. The system returns an `echeckVoidResponse` Response Code of “360” and a message of “**No transaction found with specified littleTxnId**”.
5. After you complete this test, go to the next test.

2.4.4 Testing Token Transactions

You can obtain tokens in two ways. The first method is explicit registration using the `registerTokenRequest` transaction. The second method is implicit registration, which is achieved by submitting the card or account information (for eChecks) in a normal payment transaction. This section provides test data sets using both methods for both credit card and eCheck tokenization.

Perform this test only if you plan to use the Litle & Co. Vault feature.

NOTE: Because the Certification environment is a test environment, Litle & Co. periodically purges the systems of old data. As a result of the purge operation, tokens you create in the Certification environment may no longer exist at a future time. For additional information see [Data Retention Policy](#) on page 45.

Also, the test data does not include values for all elements. You should use appropriate values for all elements as required to create a properly structured LitleXML request.

To test explicit Token Registration transactions:

1. Verify that your LitleXML template is coded correctly for this transaction type (refer to [registerTokenRequest](#) on page 380.)
2. To test credit card tokenization, submit `registerTokenRequest` transactions using the data for Order Ids 50 through 52 from [Table 2-7](#).
3. If you also use eCheck transactions and have elected to tokenize eCheck account numbers, submit `registerTokenRequest` transactions using the data for Order Ids 53 and 54 from [Table 2-7](#); otherwise, skip those two tests.
4. Verify that your `registerTokenResponse` values match those shown in the Key Response Elements section of [Table 2-7](#). The complete token values are not defined in the table, because the system generates the tokens dynamically.
5. After completing this test, proceed to the next set of tests for implicit tokenization.

TABLE 2-7 Register Token Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
50	<accountNumber>	4457119922390123	<litleToken> <bin> <type> <response> <message>	xxxxxxxxxxxx0123 445711 VI 801 Account number was successfully registered
51	<accountNumber>	4457119999999999	<litleToken> <response> <message>	none returned 820 Credit card number was invalid
52	<accountNumber>	4457119922390123	<litleToken> <bin> <type> <response> <message>	xxxxxxxxxxxx0123 445711 VI 802 Account number was previously registered
53	<accNum> <routingNum>	1099999998 114567895	<litleToken> <type> <eCheckAccountSuffix> <response> <message>	xxxxxxxxxx EC 998 801 Account number was successfully registered
54	<accNum> <routingNum>	1022222102 1145_7895	<response> <message>	900 Invalid bank routing number

To test the submission of card data in an tokenized environment using Authorization transactions, as well as the submission of tokens in transactions, do the following:

1. Verify that your LittleXML template is coded correctly for this transaction type (refer to [Authorization Transactions](#) on page 114.)
2. Submit three `authorization` transactions using the Supplied Data Elements from Order Ids 55 through 57 from [Table 2-8](#).
3. Verify that your `authorizationResponse` values match those shown in the Key Response Elements section of [Table 2-8](#) for Order Ids 55 through 57.
4. To verify that your LittleXML template is coded correctly for the submission of tokens in `authorization` transactions, submit `authorization` transactions using the Supplied Data Elements from Order Ids 58 through 60 from [Table 2-8](#).

To test the submission of eCheck data in an tokenized environment using Authorization transactions, as well as the submission of tokens in eCheck transactions, do the following:

1. Verify that your LittleXML template is coded correctly for these transaction types as applicable (refer to [eCheck Sale Transactions](#) on page 159 and [eCheck Credit Transactions](#) on page 152).
2. Submit the four transactions, Order Ids 61 through 64, using the Supplied Data Elements from [Table 2-8](#).
3. Verify that your response values match those shown in the Key Response Elements section of [Table 2-8](#) for Order Ids 61 through 64.

TABLE 2-8 Implicit Registration Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
55	<code><amount></code> <code><type></code> <code><number></code> <code><expDate></code> <code><cardValidationNum></code>	15000 MC 5435101234510196 1112 987	<code><response></code> <code><message></code> <code><littleToken></code> <code><tokenResponseCode></code> <code><tokenMessage></code> <code><type></code> <code><bin></code>	000 Approved xxxxxxxxxxxx0196 801 Account number was successfully registered MC 543510

TABLE 2-8 Implicit Registration Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
56	<amount> <type> <number> <expDate> <cardValidationNum>	15000 MC 5435109999999999 1112 987	<<response> <message>	301 Invalid account number
57	<amount> <type> <number> <expDate> <cardValidationNum>	15000 MC 5435101234510196 1112 987	<response> <message> <littleToken> <tokenResponseCode> <tokenMessage> <type> <bin>	000 Approved xxxxxxxxxxxx0196 802 Account number was previously registered MC 543510
58	<amount> <littleToken> <expDate> <cardValidationNum>	15000 xxxxxxxxxxxx0196 1112 987 Note: Use the token returned in Order Id 49.	<littleToken> <type> <eCheckAccountSuffix> <tokenResponseCode> <tokenMessage>	xxxxxxxxxxxx EC 998 801 Account number was successfully registered
59	<amount> <littleToken> <expDate>	15000 1712990000040196 1112	<response> <message>	822 Token was not found
60	<amount> <littleToken> <expDate>	15000 1712999999999999 1112	<response> <message>	823 Token was invalid

TABLE 2-8 Implicit Registration Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
61	eCheck Sale: <accType> Checking <accNum> 1099999003 <routingNum> 114567895		<littleToken> xxxxxxxxxxxxxxxxx <tokenResponseCode> 801 <tokenMessage> Account number was successfully registered <type> EC <eCheckAccountSuffix> 003	
62	eCheck Sale: <accType> Checking <accNum> 1099999999 <routingNum> 114567895		<littleToken> xxxxxxxxxxxxxxxxx <tokenResponseCode> 801 <tokenMessage> Account number was successfully registered <type> EC <eCheckAccountSuffix> 999	
63	eCheck Credit: <accType> Checking <accNum> 1099999999 <routingNum> 214567892		<littleToken> xxxxxxxxxxxxxxxxx <tokenResponseCode> 801 <tokenMessage> Account number was successfully registered <type> EC <eCheckAccountSuffix> 999	

TABLE 2-8 Implicit Registration Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
64	eCheck Sale:		<originalTokenInfo>	(parent element)
	<accType>	Corporate	<accType>	Checking
	<accNum>	6099999993	<litleToken>	11190000001003001
	<routingNum>	211370545	<routingNum>	211370545
			<newTokenInfo>	(parent element)
			<accType>	Checking
			<litleToken>	11190000001154101
			<routingNum>	211370545
			<litleToken>	xxxxxxxxxxxxxxxxxxx
			<tokenResponseCode>	801
			<tokenMessage>	Account number was successfully registered
			<type>	EC
			<eCheckAccountSuffix>	993

NOTE: Order ID 64 returns accountUpdater information. This test allows you to test responses you might receive when a NOC exists against the eCheck account, but you submit the old account information. In this case, the system provides the old token information, but issues a new token based upon the new account information and provides it as well.

2.5 Performing the Optional Tests

This section describes data that you can use to test different response codes, messages, and AVS response codes from the Little & Co. system for American Express, Visa, MasterCard, Discover, and Diner's Club cards. You can perform these tests after completing the certification testing.

This section contains the following topics:

- [Testing AVS and Card Validation](#)
- [Testing Address Responses](#)
- [Testing Advanced AVS Response Codes](#)
- [Testing Response Reason Codes and Messages](#)
- [Testing 3DS Responses](#)
- [Testing the Prepaid Filtering Feature](#)
- [Testing the International Card Filter Feature](#)
- [Testing Automatic Account Updater](#)
- [Testing Tax Billing and Convenience Fee](#)
- [Testing the Recycling Engine](#)
- [Testing Transaction Volume Capacity](#)

2.5.1 Testing AVS and Card Validation

Use the AVS tests to test all of the possible AVS response codes that the system can produce, including those response codes that cannot be produced by varying the address and ZIP code data. For these tests the AVS response codes are independent of any address or ZIP code data that you submit.

To test AVS response codes:

1. Submit transactions using the card data in [Table 2-9](#). If you are using Card Validation, include the `cardValidationNum` element. The Card Validation test will return all possible Card Validation response codes. The response codes that are returned are independent of the card validation value that you submit any value.
2. Verify that the AVS tests return the following response:

```
<response>000</response>
<message>Approved</message>
<authCode>654321</authCode>
<avsresult>See Table 2-9</avsresult>
<cardValidationResult>See Table 2-9</cardValidationResult>
```

NOTE: For a list of all possible AVS response codes, see [AVS Response Codes](#) on page 449.
For a list of all possible card validation response codes, see [Card Validation Response Codes](#) on page 453.

TABLE 2-9 AVS and Card Validation Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
65	<type> <number>	VI 4457000300000007	<avsResult> <cardValidationResult>	00 U
66	<type> <number>	VI 4457000100000009	<avsResult> <cardValidationResult>	01 M
67	<type> <number>	VI 4457003100000003	<avsResult> <cardValidationResult>	02 M
68	<type> <number>	VI 4457000400000006	<avsResult> <cardValidationResult>	10 S
69	<type> <number>	VI 4457000200000008	<avsResult> <cardValidationResult>	11 M
70	<type> <number>	MC 5112000100000003	<avsResult> <cardValidationResult>	12 M
71	<type> <number>	MC 5112002100000009	<avsResult> <cardValidationResult>	13 M
72	<type> <number>	MC 5112002200000008	<avsResult> <cardValidationResult>	14 N
73	<type> <number>	MC 5112000200000002	<avsResult> <cardValidationResult>	20 N
74	<type> <number>	MC 5112000300000001	<avsResult> <cardValidationResult>	30 P
75	<type> <number>	MC 5112000400000000	<avsResult> <cardValidationResult>	31 U

TABLE 2-9 AVS and Card Validation Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
76	<type> <number>	DI 6011000100000003	<avsResult> <cardValidationResult>	32 S
77	<type> <number>	MC 5112000500000009	<avsResult> <cardValidationResult>	33 blank
78	<type> <number>	MC 5112000600000008	<avsResult> <cardValidationResult>	34 P

2.5.2 Testing Address Responses

Use the address tests to test different AVS responses by varying the address and ZIP code data. The address tests are intended to return a realistic AVS response code.

To test address responses:

1. Submit transactions using the data in [Table 2-10](#). This table shows credit card numbers with the correct billing address for each credit card. All transactions return an `avsResult` of 00.
2. The AVS response code depends on the Address Line 1 and ZIP Code that are passed in with the transaction. Submit additional transactions using the card data from the table, but varying the address/zip information to receive other `avsResult` codes in the response messages (see Examples below).

Example: Correct Address and Nine-Digit ZIP Code

AVS Request: 4457000600000004 95 Main St. 950221111

AVS Response: 01

Example: Incorrect Address, but Correct Five-Digit ZIP Code

AVS Request: 4457000600000004 100 Maple St 95022

AVS Response: 10

Example: Incorrect Address and ZIP Code

AVS Request: 4457000600000004 100 Maple St 02134

AVS Response: 20

For a detailed list of all possible AVS response codes, see [AVS Response Codes](#) on page 449.

TABLE 2-10 Address Test Data

Supplied Data Elements	
Element	Value
<type> <number>	VI 4457000600000004
<type> <number>	MC 5112000700000007
<type> <number>	AX 3750000100000005
<type> <number>	DI 6011000200000002
<type> <number>	VI 4457000700000003
<type> <number>	MC 5112000800000006
<type> <number>	AX 3750000200000003
<type> <number>	DI 6011000300000001

2.5.3 Testing Advanced AVS Response Codes

The Advanced AVS (AAVS) feature is an offering from American Express that allows you to check several parameters not normally covered by a standard AVS check, including name, phone, and email.

To test AAVS Response Codes:

1. Submit an Authorization transaction using the data supplied in [Table 2-12](#).
2. Verify that you handle the response correctly.
3. Enter additional transaction varying the values for name, phone, and/or email to trigger other AAVS results (see [AAVS Response Codes](#) on page 450 for other result codes). For example, if you submit a second transaction using the name Jane Doe Instead of John Doe, the AAVS result would be 011 indicating No Match for name, but Match for phone and email.

TABLE 2-11 Prepaid Filtering Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
80	<amount> <name> <addressLine1> <city> <state> <zip> <country> <email> <phone> <type> <number> <expDate>	12523 John Doe 95 Main St. Palo Alto CA 950221111 US test@test.com 6178675309 AX 341234567890127 1113	<advancedAVSResults>	111

2.5.4 Testing Response Reason Codes and Messages

Use the data as shown in this section to test Response Reason Codes and Messages.

NOTE: If you submit account numbers not specified in the tables, you will receive the following response:

```
<response>000</response>
<message>Approved</message>
<authCode>123457</authCode>
<avsResult>00</avsResult>
```

To test Response Codes and Messages:

1. Submit transactions using the data in [Table 2-12](#).
2. Verify that you handle the response correctly. The responses are as indicated in [Table 2-12](#) regardless of the address sent in with the transaction.

NOTE: The messages listed are samples of messages that the system can return. Since the messages are subject to change at any time, you should use them only for human readability purposes and not for coding purposes. Always code to the response codes, since these do not change.

- For a list of all possible response reason codes, see [Payment Transaction Response Codes](#) on page 434.
- For a list of all possible AVS response codes, see [AVS Response Codes](#) on page 449.

TABLE 2-12 AVS and Card Validation Test Data

Supplied Data Elements		Key Response Elements	
Element	Value	Element	Value
<type> <number>	VI 4457000800000002	<response> <message>	000 Approved
<type> <number>	VI 4457000900000001	<response> <message>	000 Approved
<type> <number>	VI 4457001000000008	<response> <message>	000 Approved
<type> <number>	MC 5112000900000005	<response> <message>	000 Approved

TABLE 2-12 AVS and Card Validation Test Data

Supplied Data Elements		Key Response Elements	
Element	Value	Element	Value
<type> <number>	AX 3750000300000001	<response> <message>	121 Call AMEX
<type> <number>	DI 6011000400000000	<response> <message>	123 Call Discover
<type> <number>	VI 4457001200000006	<response> <message>	120 Call Issuer
<type> <number>	VI 4457001300000005	<response> <message>	120 Call Issuer
<type> <number>	VI 4457001400000004	<response> <message>	120 Call Issuer
<type> <number>	MC 5112001000000002	<response> <message>	101 Issuer Unavailable
<type> <number>	VI 4457001900000009	<response> <message>	321 Invalid Merchant
<type> <number>	VI 4457002000000006	<response> <message>	303 Pick Up Card
<type> <number>	VI 4457002100000005	<response> <message>	110 Insufficient Funds
<type> <number>	VI 4457002200000004	<response> <message>	120 Call Issuer
<type> <number>	AX 3750000500000006	<response> <message>	110 Insufficient Funds
<type> <number>	VI 4457002300000003	<response> <message>	349 Do Not Honor
<type> <number>	VI 4457002500000001	<response> <message>	340 Invalid Amount
<type> <number>	MC 5112001600000006	<response> <message>	301 Invalid Account Number

TABLE 2-12 AVS and Card Validation Test Data

Supplied Data Elements		Key Response Elements	
Element	Value	Element	Value
<type> <number>	MC 5112001700000005	<response> <message>	301 Invalid Account Number
<type> <number>	MC 5112001800000004	<response> <message>	321 Invalid Merchant
<type> <number>	VI 4457002700000009	<response> <message>	101 Issuer Unavailable
<type> <number>	MC 5112001900000003	<response> <message>	305 Expired Card
<type> <number>	VI 4457002800000008	<response> <message>	322 Invalid Transaction
<type> <number>	VI 4457002900000007	<response> <message>	350 Generic Decline
<type> <number>	VI 4457003000000004	<response> <message>	101 Issuer Unavailable
<type> <number>	MC 5112002000000000	<response> <message>	101 Issuer Unavailable
<type> <number>	VI 4457000100000000	<response> <message>	301 Invalid Account Number
<type> <number>	VI 4457000200000008	<response> <message>	320 Invalid Expiration Date

2.5.5 Testing 3DS Responses

The cardholder authentication value should only be included by merchants who support 3DS (3 Domain Secure) electronic commerce transactions. Your systems must be in compliance with the Verified by Visa or MasterCard Secure Code implementations of 3DS.

To test 3DS responses:

1. Submit Authorization transactions or Sale transactions using the data in [Table 2-13](#). For all cards, set the cardholder authentication value within <cardholderAuthentication> section to the following base64 encoded string:

BwABBJQ1AgAAAAgJDUCAAAAAA=

The response from a 3DS test will be the same as an Authorization or Sale response, except the `<authenticationResult>` tag will be included in the response within the `<fraudResult>` section.

TABLE 2-13 3DS Test Data

Supplied Data Elements		Key Response Elements	
Element	Value	Element	Value
<code><type></code> <code><number></code>	VI 4457010200000015	<code><authenticationResult></code>	0
<code><type></code> <code><number></code>	VI 4457010200000023	<code><authenticationResult></code>	1
<code><type></code> <code><number></code>	VI 4457010200000031	<code><authenticationResult></code>	2
<code><type></code> <code><number></code>	VI 4457010200000049	<code><authenticationResult></code>	3
<code><type></code> <code><number></code>	VI 4457010200000056	<code><authenticationResult></code>	4
<code><type></code> <code><number></code>	VI 4457010200000064	<code><authenticationResult></code>	5
<code><type></code> <code><number></code>	VI 4457010200000072	<code><authenticationResult></code>	6
<code><type></code> <code><number></code>	VI 4457010200000080	<code><authenticationResult></code>	7
<code><type></code> <code><number></code>	VI 4457010200000098	<code><authenticationResult></code>	8
<code><type></code> <code><number></code>	VI 4457010200000106	<code><authenticationResult></code>	9
<code><type></code> <code><number></code>	VI 4457010200000114	<code><authenticationResult></code>	A
<code><type></code> <code><number></code>	VI 4457010200000122	<code><authenticationResult></code>	B

TABLE 2-13 3DS Test Data

Supplied Data Elements		Key Response Elements	
Element	Value	Element	Value
<type> <number>	VI 4457010200000130	<authenticationResult>	C
<type> <number>	VI 4457010200000148	<authenticationResult>	D
<type> <number>	MC 5112010200000001	<authenticationResult>	N/A

2.5.6 Testing the Prepaid Filtering Feature

Complete this test only if you are planning on using the Litle & Co. Prepaid Filtering Feature and are using schema version 8.3 or above.

To test the Prepaid Filtering feature:

1. Submit authorization transactions using the values provided in Supplied Data Elements of [Table 2-14](#).
2. Verify that your response values match those shown in the Key Response Elements section of [Table 2-14](#).
3. After you complete this test, go to the next test.

TABLE 2-14 Prepaid Filtering Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
filterPrepaidAX	<amount> <orderSource> <name> <addressLine1> <city> <state> <zip> <country> <email> <phone> <type> <number> <expDate> <prepaid>	9100 recurring John Doe 10 Main St. San Jose CA 95032 US jdoe@phoenixProcessing.com 7812701111 AX 375001020000100 1113 true	<response> <message> <authCode> <avsResult>	309 Restricted Card - Prepaid Card Filtering Service AXPREP 02
filterPrepaidVI	<amount> <orderSource> <name> <addressLine1> <city> <state> <zip> <country> <email> <phone> <type> <number> <expDate> <prepaid>	9100 recurring John Doe 10 Main St. San Jose CA 95032 US jdoe@phoenixProcessing.com 7812701111 VI 4100200010001474 1113 true	<response> <message> <authCode> <avsResult>	309 Restricted Card - Prepaid Card Filtering Service none 34

2.5.7 Testing the International Card Filter Feature

Complete this test only if you are planning on using the Litle & Co. International Card Filtering Feature and are using schema version 8.3 or above.

To test the International Card Filtering feature:

1. Submit authorization transactions using the values provided in Supplied Data Elements of [Table 2-15](#).
2. Verify that your response values match those shown in Key Response Elements section of [Table 2-15](#).
3. After you complete this test, go to the next test.

TABLE 2-15 International Filtering Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
filterInternational1	<amount>	9100	<response>	312
	<orderSource>	recurring	<message>	Restricted Card - International Card Filtering Service
	<name>	John Doe	<avsResult>	34
	<addressLine1>	10 Main St.		
	<city>	San Jose		
	<state>	CA		
	<zip>	95032		
	<country>	US		
	<email>	jdoe@phoenixProcessing.com		
	<phone>	7812701111		
	<type>	VI		
	<number>	4100200309950001		
	<expDate>	1113		

TABLE 2-15 International Filtering Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
filterInternational2	<amount>	9100	<response>	000
	<orderSource>	recurring	<message>	Approved
	<name>	John Doe	<authCode>	123457
	<addressLine1>	10 Main St.	<avsResult>	00
	<city>	San Jose		
	<state>	CA		
	<zip>	95032		
	<country>	US		
	<email>	jdoe@phoenixProcessing.com		
	<phone>	7812701111		
	<type>	VI		
	<number>	4100200309950001		
	<expDate>	1113		
	<international>	false		

2.5.8 Testing Automatic Account Updater

To test Automatic Account Updater, you submit a normal Authorization transaction. The certification system returns an Authorization response that includes Account Update information. You should verify that you correctly parse the update information.

NOTE: You can also perform the tests in this section using Sale transactions instead of Authorization transactions.

To test the Automatic Account Updater service:

1. Submit authorization transactions using the values provided in Supplied Data Elements of [Table 2-16](#).
2. Verify that your response values match those shown in Key Response Elements section of [Table 2-16](#).
3. If you have coded to receive Extended Response Codes, proceed to the next test.

TABLE 2-16 Automatic Account Updater Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
100	<amount>	10000	<originalCardInfo>	(parent element)
	<type>	VI	<type>	VI
	<number>	4457000300000007	<number>	4457000300000007
	<expDate>	0115	<expDate>	0115
			<newCardInfo>	(parent element)
			<type>	MC
			<number>	5112000100000003
			<expDate>	0115
101	<amount>	10000	<originalCardInfo>	(parent element)
	<type>	DI	<type>	DI
	<number>	6500102087026221	<number>	6500102087026221
	<expDate>	0115	<expDate>	0115
			<newCardInfo>	(parent element)
			<type>	DI
			<number>	6011102077026225
			<expDate>	0115

2.5.8.1 Testing Automatic Account Updater Extended Response Codes

To test the Automatic Account Updater Extended Response Codes feature:

NOTE: You are required to code to LitleXML schema version 8.5 or above to receive the <extendedCardResponse> child of <accountUpdater>.

1. Submit `authorization` transactions using the values provided in Supplied Data Elements of [Table 2-17](#).
2. Verify that the response values match those shown in Key Response Elements section of [Table 2-17](#) and that your systems parse the data correctly. The second test case does not include account repair information only the Extended Response Code.

TABLE 2-17 Automatic Account Updater Extended Response Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
102	<amount>	10000	<originalCardInfo>	(parent element)
	<type>	MC	<type>	MC
	<number>	5112000101110009	<number>	5112000101110009
	<expDate>	1199	<expDate>	1199
			<newCardInfo>	(parent element)
			<type>	VI
			<number>	4457000302200001
103			<expDate>	1199
			<extendedCardResponse>	(parent element)
			<code>	501
			<message>	The account was closed.
103	<amount>	10000	<extendedCardResponse>	(parent element)
	<type>	VI	<code>	501
	<number>	4457000301100004	<message>	Contact the cardholder for updated information.
	<expDate>	1199		

2.5.8.2 Testing Automatic Account Updater for Tokenized Merchants

If you are a tokenized merchant using the Automatic Account Updater service, you can test this service using the card information provided in [Table 2-16](#). In this case you will receive an original and new token in the <accountUpdater> section of the Authorization response message (see [accountUpdater Structure - Credit Cards \(tokenized Merchant\)](#) on page 196).

2.5.9 Testing Tax Billing and Convenience Fee

This test applies only to merchants with MCC 9311.

To test Tax Billing and Convenience Fee transactions:

1. Submit `authorization` transactions using the values provided in Supplied Data Elements of [Table 2-18](#). Note: The second transactions omits the `<taxType>` element.
2. Verify that the system returns a response code of 000 - Approved for the first transaction and response code of 852 - Invalid Tax Billing for the second.

TABLE 2-18 Tax Billing and Convenience Fee Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
MCC9311Test	<code><amount></code> <code><type></code> <code><number></code> <code><expDate></code> <code><taxType></code>	3000 VI 4457010200000247 1113 fee	<code><response></code> <code><message></code>	000 Approved
MCC9311Test2	<code><amount></code> <code><type></code> <code><number></code> <code><expDate></code>	3000 VI 4457010200000247 1113	<code><response></code> <code><message></code>	851 Invalid Tax Billing

2.5.10 Testing the Recycling Engine

The Certification test cases for the Little Recycling Engine serve two purposes. First, you use the test transactions to verify your handling of the responses you receive if you submit additional Authorization transactions for a declined auth being handled by the engine. Second, you can verify your process for retrieving and processing recycling completion files via sFTP.

To test the Recycling Engine:

NOTE: You can also perform these tests using `sale` transactions instead of `authorization` transactions.

1. Submit `authorization` (or `sale`) transactions using the values provided for Test Case 1 and 2 in the Supplied Data Elements of [Table 2-19](#).

2. One day later resubmit the same transactions. Verify that the response reason codes match the values provided in the Key response Elements section based upon the schema version you are using.
3. Access your FTP directory and retrieve the response file. The FTP directory path is /prod/certftp/merchantShortName/merchantShortName/outbound/. Your Little Implementation Consultant will provide the value for "merchantShortName". The file name will have the following format:recycling.merchantId.MMddyyyy.sessionId.response. For example, the name of the example file shown below would be (assuming date of 12/02/11): recycling.042400.12022011.82818202122567706.response

Since the recycling Engine has not yet received approval for the submitted transactions, the retrieved file will be an empty batch similar to the following:

```
<littleResponse version="8.9" xmlns="http://www.little.com/schema" response="0"
message="Valid Format" littleSessionId="82818202122567706">

<batchResponse littleBatchId="82818202122569694" merchantId="042400">

</batchResponse>

</littleResponse>
```

4. One day later, after the time provided by your Implementation Consultant, resubmit the same transactions. Verify that the response reason codes match the values provided in the Key response Elements section based upon the schema version you are using.
5. For the transaction with `orderId = recycleVIcredit2`, submit a `capture` transaction using the `littleTxnId` returned in the approved Authorization response message. The `capture` transaction will be approved.
6. Access your FTP directory and retrieve the response file. This response file will contain two transactions. One will be the approved response for the first test case, while the response for the second transaction will contain a 373 response code. The 373 response code indicates the recycling pattern of the engine has been exhausted. You can continue to recycle this authorization manually and it will be passed through to the network (response will be 110 - Insufficient Funds) until day 30. After day 30 the engine will mark it as a new transaction and mark it for automatic recycling.

NOTE: In production you are not required to resubmit the Authorization transactions (Steps 2 and 4). These tests are performed in the Certification environment to demonstrate the transaction intercept function and for you to test your handling of the 372 and 373 response codes. Normally, you will receive the approvals or final decline responses via the Batch files pushed daily to the FTP site.

TABLE 2-19 Recycling Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
(Test Case 1) recycleVlcredit2	<amount> <orderSource> <type> <number> <expDate>	127 mailorder VI 4457012400000027 1220	<response> <message>	110 Insufficient Funds
recycleVlcredit2 (2nd attempt; initial + 1 day)	<amount> <orderSource> <type> <number> <expDate>	127 mailorder VI 4457012400000027 1220	If V8.6 or greater: <response> <message> If V8.5 or below: <response> <message>	372 Soft Decline - Auto Recycling in Progress 110 Insufficient Funds
recycleVlcredit2 (3rd attempt; initial + 2 day)	<amount> <orderSource> <type> <number> <expDate>	127 mailorder VI 4457012400000027 1220	<response> <message>	000 Approved
(Test Case 2) recycleVlcredit3	<amount> <orderSource> <type> <number> <expDate>	135 mailorder VI 4457012400000035 1220	<response> <message>	110 Insufficient Funds
recycleVlcredit3 (2nd attempt; initial + 1 day)	<amount> <orderSource> <type> <number> <expDate>	135 mailorder VI 4457012400000035 1220	If V8.6 or greater: <response> <message> If V8.5 or below: <response> <message>	372 Soft Decline - Auto Recycling in Progress 110 Insufficient Funds

TABLE 2-19 Recycling Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
recycleVlcredit3 (3rd attempt; initial + 2 day)	<amount>	135	If V8.6 or greater:	
	<orderSource>	mailorder	<response>	373
	<type>	VI	<message>	Hard Decline - Auto Recycling Complete
	<number>	4457012400000035	If V8.5 or below:	
	<expDate>	1220	<response>	110
			<message>	Insufficient Funds

2.5.10.1 Testing Recycling Engine Cancellation

You use an `authReversal` transaction to halt the automatic recycling of an authorization. For a sale transaction, use a `void` transaction to halt recycling.

NOTE: You can perform this test either after completing the Recycling Engine test or prior to starting that test.

To test recycling cancellation

1. Submit `authorization` transaction using the values provided for Test Case 1 and a sale transaction the values provided for Test Case 2 in the Supplied Data Elements of [Table 2-19](#).
2. After receiving the decline messages, submit an `authReversal` transaction using the `littleTxnId` returned in the response message for Test Case 1. You should receive an `authReversalResponse` with a response code of 000 - Approved.
3. After receiving the decline messages, submit a `void` transaction using the `littleTxnId` returned in the response message for Test Case 2. You should receive an `voidResponse` with a response code of 000 - Approved.

2.5.11 Testing Transaction Volume Capacity

Volume testing is useful if you plan to send large files. This is an optional test you can perform during certification testing. Volume testing enables you to verify how many transactions (the number of requests and responses) you can process within a specific time frame.

Little & Co. recommends you submit transactions for a 15-minute time interval. Submit the approximate number of transactions that you anticipate to be normal volume for any 15-minute period. You can send in any valid transaction data; the actual data you send will not be verified.



LITLEXML TRANSACTION EXAMPLES

This chapter contains information and examples concerning the structure of LitleXML transaction messages. Where differences exist between the structure of Batch and Online transactions, the sections present examples of both structures.

This chapter discusses the following topics:

- [Overview of Online and Batch Processing Formats](#)
- [Transaction Types and Examples](#)

3.1 Overview of Online and Batch Processing Formats

There are two methods of submitting payment transactions using the LitleXML format: Online (one transaction at a time), or Batch. This section provides a high level overview of the request and response structures used for each submission type.

3.1.1 Batch Process Format

Each Batch transmission you send to Litle & Co. is considered to be a single request. You can think of the entire Batch request as a session composed of one or more individual batches, each containing one or more transactions. You can also use a Batch as a request for retrieval of the response for a previously processed session. Each request results in a response transmission sent from Litle & Co. to you.

Batch processing supports the following transaction types;

- Authorizations
- Authorization Reversal
- Captures
- Capture Given Auth
- Force Capture
- Sale Transactions
- Credit Transactions
- eCheck Sale Transactions
- eCheck Credit Transactions
- eCheck Redeposit Transactions
- eCheck Verification Transactions
- RFR Batch Transactions (Batch Only)

For more information about Batch processing, see [Batch Transaction Processing](#) on page 5.

3.1.1.1 Supported Communication Protocols

Litle & Co. supports the following communication protocols for Batch processing. Litle & Co. recommends using either Encrypted FTP or sFTP for the transmission of Batch files.

- HTTPS POST
- Encrypted FTP (PGP or GPG key encryption)
- sFTP

For additional information concerning the recommended transmission methods, see [Transferring Batch Files](#) on page 48.

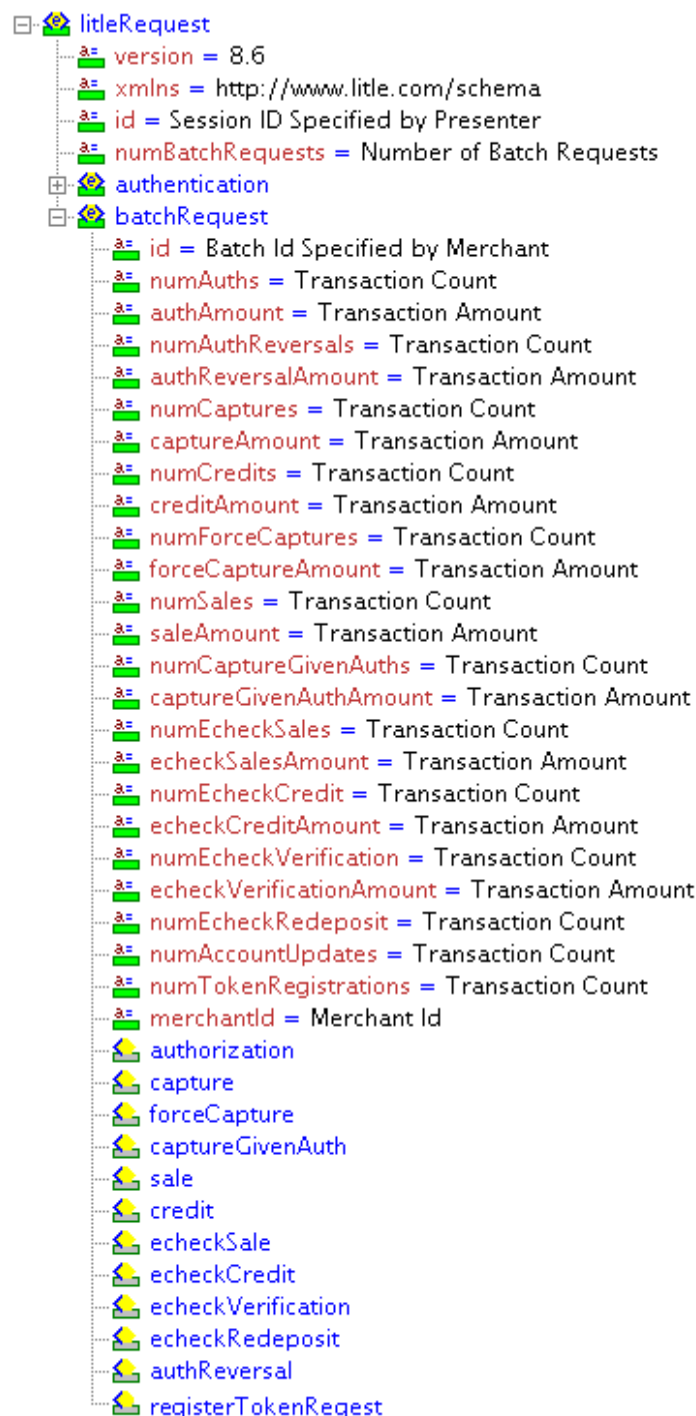
3.1.1.2 Batch Processing Request Format

The Batch processing request is made up of the following elements:

- Header information - one `<littleRequest>` element
- Merchant authentication information - one `<authentication>` element
- Batch information - one or more `<batchRequest>` elements
- Payment transactions (for example, an Authorization, Capture, Credit, etc.) - at least one per `<batchRequest>` element

[Figure 3-1](#) illustrates the required structure of a Batch Request.

NOTE: Each of the `num` and `amount` attributes used in a `batchRequest` defaults to 0 (zero) if not specified in the request. In any of the `amount` fields (`authAmount`, `captureAmount`, etc.), the value is an implied decimal (for example 500=5.00).

FIGURE 3-1 Batch Request XML Format

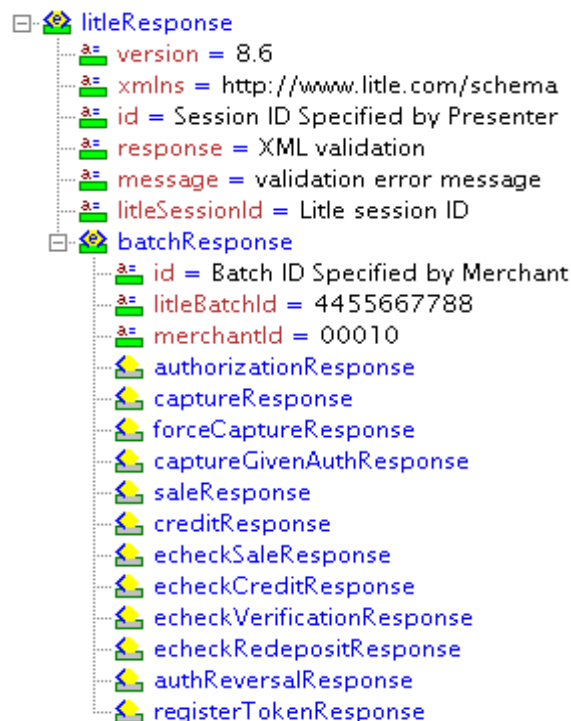
3.1.1.3 Batch Processing Response Format

The Batch processing response is composed of the following elements:

- Header information - one `<littleResponse>` element
- One or more `<batchResponse>` elements - contains payment transactions response.
- At least one payment transaction response (for example, an Authorization response, Capture response, Credit response, etc.).

Figure 3-2 illustrates the structure of a Batch Response

FIGURE 3-2 Batch Response XML Format



NOTE: For information on the XML Validation response and message attributes, please refer to [XML Validation Error Messages](#) on page 454.

3.2 Online Processing Format

Each Online request you send to Litle & Co. is a single transaction. Litle processes Online transactions upon receipt and returns a response file.

Online processing supports the following transaction types:

- Authorizations
- Authorization Reversal
- Capture
- Capture Given Auth
- Force Capture
- Sale Transactions
- Credit Transactions
- eCheck Sale Transactions
- eCheck Credit Transactions
- eCheck Redeposit Transactions
- eCheck Verification Transactions
- eCheck Void Transactions (Online Only)
- Void Transactions (Online Only)

3.2.1 Supported Communication Protocols

Litle & Co. supports the following communication protocol for Online processing is HTTPS POST.

For additional information concerning the recommended transmissions methods, see [Transferring Online Files](#) on page 49.

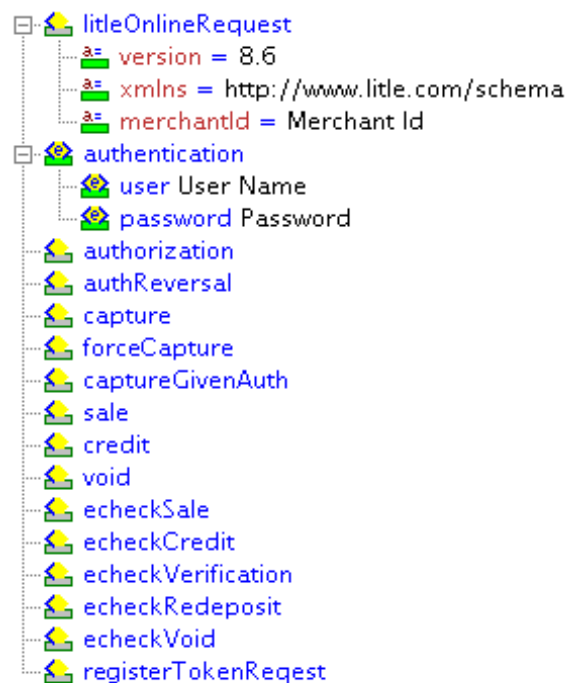
3.2.2 Online Processing Request Format

The Online processing request is made up of the following elements:

- Header information - one `<littleOnlineRequest>` element
- Merchant authentication information - one `<authentication>` element
- Payment transaction - one payment transaction

Figure 3-3 illustrates the structure of an Online Request.

FIGURE 3-3 Online Request XML Format



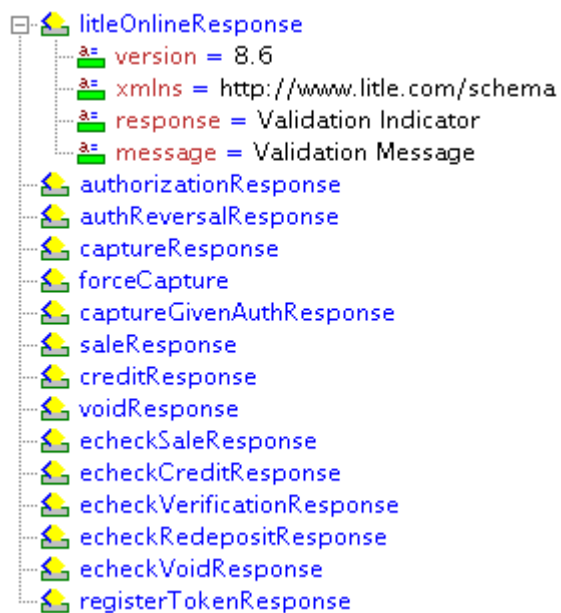
3.2.3 Online Processing Response Format

An Online processing response is composed of the following elements:

- Header information - one `<littleOnlineResponse>`
- Payment transaction - one payment transaction

Figure 3-4 illustrates the structure of an Online Response.

FIGURE 3-4 Online Response XML Format



NOTE: For information on the XML Validation response and message attributes, please refer to [XML Validation Error Messages](#) on page 454.

3.3 Transaction Types and Examples

This section presents structural information of each transaction type for both Online and Batch submission methods. The structural information is followed by one or more examples of the LittleXML transaction. Each structural example shows the parent and all child elements, but does not show grandchildren. The LittleXML examples do show child elements to multiple levels.

The element names in the structural examples provide links to the element definitions in Chapter 4.

NOTE: The XML examples in this section are intended to present typical LittleXML transactions. The examples may not include every possible element for a particular transaction type. When coding your XML, always consult the LittleXML schema files for information concerning all available elements.

This section contains examples of the following transaction types:

- [Authorization Transactions](#)
- [Authorization Reversal Transactions](#)
- [Capture Transactions](#)
- [Capture Given Auth Transactions](#)
- [Credit Transactions](#)
- [eCheck Credit Transactions](#)
- [eCheck Redeposit Transactions](#)
- [eCheck Sale Transactions](#)
- [eCheck Verification Transactions](#)
- [eCheck Void Transactions \(Online Only\)](#)
- [Force Capture Transactions](#)
- [Register Token Transactions](#)
- [RFR Transactions \(Batch Only\)](#)
- [Sale Transactions](#)
- [Void Transactions \(Online Only\)](#)

3.3.1 Authorization Transactions

The Authorization transaction enables you to confirm that a customer has submitted a valid payment method with their order and has sufficient funds to purchase the goods or services they ordered.

The lifespan of an authorization varies according to the payment type being used, as shown in [Table 3-1](#). During the lifespan, you can use a valid authorization multiple times as needed.

NOTE: To submit an AVS Only request, submit an Authorization request with the `<amount>` element set to 000. The response is identical to an Authorization response message.

TABLE 3-1 Lifespan of a Payment Authorization

Payment Type	Lifespan of Authorization
American Express	7 days
Bill Me Later	30 days by default; for more information about Bill Me Later authorizations, see the <i>Litle & Co. Bill Me Later Integration Guide</i> .
Discover	30 days
MasterCard	7 days
PayPal	29 days total; Litle & Co. recommends three days. For more information about PayPal authorizations, see the <i>Litle & Co. PayPal Integration Guide</i> .
Visa	7 days

This section describes the format you must use for an Authorization request, as well as the format of the Authorization Response you receive from Litle & Co.

3.3.1.1 Authorization Request Structure

You must structure an Authorization request as shown in the following examples. The structure of an Authorization request is identical for either an Online or a Batch submission.

```
<authorization id="Authorization Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Authorization Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <customerInfo>
```

```

    <billToAddress>
    <shipToAddress>
    [ <card> | <paypal> | <paypage> | <token>]
    <billMeLaterRequest>
    <cardholderAuthentication>
    <processingInstructions>
    <pos>
    <customBilling>
    <taxType>payment or fee</taxType>
    <enhancedData>
    <amexAggregatorData>
    <allowPartialAuth>
    <healthcareIIAS>
    <filtering>
    <merchantData>
    <recyclingRequest>
  </authorization>

```

Example: Batch Authorization Request - Card Not Present

The example below shows a batch request with a single card-not-present Authorization request. If the batch included additional Authorization requests, each would have its own `<authorization>` element with all applicable attributes and child elements. Also, the `numAuths` attribute of the `<batchRequest>` element would increment for each additional `<authorization>` element and the `authAmount` attribute would increase by the new amounts from each authorization.

```

<littleRequest version="8.10" xmlns="http://www.little.com/schema"
  numBatchRequests = "1">
  <authentication>
    <user>XMLTESTP7</user>
    <password>password</password>
  </authentication>
  <batchRequest numAuths="1" authAmount="2500" merchantId="000902">
    <authorization id="test1" reportGroup="core" customerId="test1">
      <orderId>visa_test1</orderId>
      <amount>2500</amount>
      <orderSource>telephone</orderSource>
      <billToAddress>
        <name>John Doe</name>
        <addressLine1>15 Main Street</addressLine1>

```

```

    <city>San Jose</city>
    <state>CA</state>
    <zip>95032-1234</zip>
    <country>USA</country>
    <phone>9782750000</phone>
    <email>jdoe@litle.com</email>
  </billToAddress>
  <shipToAddress>
    <name>Jane Doe</name>
    <addressLine1>15 Main Street</addressLine1>
    <city>San Jose</city>
    <state>CA</state>
    <zip>95032-1234</zip>
    <country>USA</country>
    <phone>9782750000</phone>
    <email>jdoe@litle.com</email>
  </shipToAddress>
  <card>
    <type>VI</type>
    <number>4005550000081019</number>
    <expDate>1110</expDate>
  </card>
  <customBilling>
    <phone>8009990001</phone>
    <descriptor>bdi*001</descriptor>
  </customBilling>
  <allowPartialAuth>true</allowPartialAuth>
</authorization>
</batchRequest>
</litleRequest>

```

Example: Batch Authorization Request - Card Present

The following example contains two Authorization requests, each defined in its own <authorization> element. The first is a card present transaction, which uses the <track> child of the <card> element.

```

<litleRequest version="8.10" xmlns="http://www.litle.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>

```



```
<batchRequest id="01234567" numAuths="2" authAmount="68336"
merchantId="100">
  <authorization id="AX54321678" reportGroup="RG27">
    <orderId>12z58743y1</orderId>
    <amount>12522</amount>
    <orderSource>retail</orderSource>
    <billToAddress>
      <zip>95032</zip>
    </billToAddress>
    <card>
      <track>%B40000001^Doe/JohnP^06041...?;40001=0604101064200?</track>
    </card>
    <pos>
      <capability>magstripe</capability>
      <entryMode>completeread</entryMode>
      <cardholderId>signature</cardholderId>
    </pos>
  </authorization>
  <authorization id="AX54325432" reportGroup="RG12">
    <orderId>12z58743y7</orderId>
    <amount>55814</amount>
    <orderSource>retail</orderSource>
    <billToAddress>
      <zip>01854</zip>
    </billToAddress>
    <card>
      <type>VI</type>
      <number>4005550000081019</number>
      <expDate>0911</expDate>
    </card>
    <pos>
      <capability>keyedonly</capability>
      <entryMode>keyed</entryMode>
      <cardholderId>directmarket</cardholderId>
    </pos>
    <allowPartialAuth>true</allowPartialAuth>
  </authorization>
</batchRequest>
</littleRequest>
```

Example: Online Authorization Request

NOTE: The example below uses 3dsAuthenticated as the <orderSource> value. If you submit the wrong <orderSource> value, Little returns the response code 370 - Internal System Error - Contact Little.

Also, the values for the <authenticationValue> and <authenticationTransactionId> elements in the example below have been truncated.

```
<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <authorization id="834262" reportGroup="ABC Division" customerId="038945">
    <orderId>65347567</orderId>
    <amount>40000</amount>
    <orderSource>3dsAuthenticated</orderSource>
    <billToAddress>
      <name>John Smith</name>
      <addressLine1>100 Main St</addressLine1>
      <city>Boston</city>
      <state>MA</state>
      <zip>12345</zip>
      <email>jsmith@someaddress.com</email>
      <phone>555-123-4567</phone>
    </billToAddress>
    <card>
      <type>VI</type>
      <number>4000000000000001</number>
      <expDate>1209</expDate>
      <cardValidationNum>555</cardValidationNum>
    </card>
    <cardholderAuthentication>
      <authenticationValue>BwABBJQ1gJDUCAAAAAA=</authenticationValue>
      <authenticationTransactionId>gMV75TmjAgk=</authenticationTransactionId>
    </cardholderAuthentication>
  </authorization>
</littleOnlineRequest>
```

Example: Authorization Request using token Element

The example below uses the following token related elements (click name to jump to element definition): [token](#) and [littleToken](#).

```
<authorization id="99999" customerId="444" reportGroup="RG1">
  <orderId>F12345</orderId>
  <amount>15000</amount>
  <orderSource>telephone</orderSource>
  <billToAddress>
    <name>John Doe</name>
    <addressLine1>15 Main Street</addressLine1>
    <city>San Jose</city>
    <state>CA</state>
    <zip>95032-1234</zip>
    <country>USA</country>
    <phone>9782750000</phone>
    <email>jdoe@little.com</email>
  </billToAddress>
  <shipToAddress>
    <name>Jane Doe</name>
    <addressLine1>15 Main Street</addressLine1>
    <city>San Jose</city>
    <state>CA</state>
    <zip>95032-1234</zip>
    <country>USA</country>
    <phone>9782750000</phone>
    <email>jdoe@little.com</email>
  </shipToAddress>
  <token>
    <littleToken>1111000101039449</littleToken>
    <expDate>1112</expDate>
    <cardValidationNum>987</cardValidationNum>
  </token>
</authorization>
```

3.3.1.2 Authorization Response Structure

An Authorization response has the following structure. The response message is identical for Online and Batch transactions except Online includes the `<postDate>` element.

```
<authorizationResponse id="Authorization Id" reportGroup="UI Report
Group" customerId="Customer Id">
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date transaction posted</postDate> (Online Only)
  <message>Response Message</message>
  <authCode>Approval Code</authCode>
  <approvedAmount>Approved amount for partial Auth<approvedAmount>
  <accountInformation>
  <accountUpdater>
  <fraudResult>
  <billMeLaterResponseData>
  <tokenResponse> (for Tokenized merchants submitting card data)
  <enhancedAuthResponse>
  <accountUpdater>
  <recycling> (included for declined Auths if featrure is enabled)
</authorizationResponse>
```

Example: Batch Authorization Response

The example below shows a batch Authorization response that contains two transactions.

```
<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
  response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <authorizationResponse id="AX54321678" reportGroup="RG27">
      <littleTxnId>84568456</littleTxnId>
      <orderId>12z58743y1</orderId>
      <response>000</response>
      <responseTime>2011-03-01T10:24:31</responseTime>
      <message>Approved</message>
      <authCode>123456</authCode>
      <fraudResult>
        <avsResult>00</avsResult>
      </fraudResult>
    </authorizationResponse>
  </batchResponse>
</littleResponse>
```

```

    </fraudResult>
</authorizationResponse>
<authorizationResponse id="AX54325432" reportGroup="RG12">
  <littleTxnId>84568457</littleTxnId>
  <orderId>12z58743y7</orderId>
  <response>000</response>
  <responseTime>2011-03-01T10:24:31</responseTime>
  <message>Approved</message>
  <authCode>123456</authCode>
  <fraudResult>
    <avsResult>00</avsResult>
    <authenticationResult>2</authenticationResult>
  </fraudResult>
  <enhancedAuthResponse>
    <fundingSource>
      <type>PREPAID</type>
      <availableBalance>5000</availableBalance>
      <reloadable>NO</reloadable>
      <prepaidCardType>GIFT</prepaidCardType>
    </fundingSource>
  </enhancedAuthResponse>
</authorizationResponse>
</batchResponse>
</littleResponse>

```

Example: Online Authorization Response

NOTE: The online response format contains a `<postDate>` element, which indicates the date the financial transaction will post (specified in YYYY-MM-DD format).

```

<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <authorizationResponse id="834262" reportGroup="ABC Division">
    <littleTxnId>969506</littleTxnId>
    <orderId>65347567</orderId>
    <response>000</response>
    <responseTime>2011-07-25T15:13:43</responseTime>
    <postDate>2011-07-25</postDate>
    <message>Approved</message>
    <authCode>123457</authCode>
    <fraudResult>

```

```

    <avsResult>00</avsResult>
    <cardValidationResult>N</cardValidationResult>
    <authenticationResult>2</authenticationResult>
  </fraudResult>
  <enhancedAuthResponse>
    <fundingSource>
      <type>PREPAID</type>
      <availableBalance>0</availableBalance>
      <reloadable>YES</reloadable>
      <prepaidCardType>TEEN</prepaidCardType>
    </fundingSource>
  </enhancedAuthResponse>
</authorizationResponse>
</littleOnlineResponse>

```

Example: Authorization Response for Tokenized Merchant Sending Card Data

If a tokenized merchant submits card data in the Authorization request, the response includes the `tokenResponse` element. The example below is a response for an Online request (`littleOnlineResponse` element not shown); therefore, it includes the `postDate` element.

```

<authorizationResponse id="99999" reportGroup="RG1" customerId="444">
  <littleTxnId>21200000001108</littleTxnId>
  <orderId>F12345</orderId>
  <response>000</response>
  <responseTime>2011-10-08T21:38:32</responseTime>
  <postDate>2011-10-08</postDate>
  <message>Approved</message>
  <authCode>270005</authCode>
  <fraudResult>
    <avsResult>11</avsResult>
    <cardValidationResult>P</cardValidationResult>
  </fraudResult>
  <tokenResponse>
    <littleToken>1111100100240005</littleToken>
    <tokenResponseCode>801</tokenResponseCode>
    <tokenMessage>Account number was successfully registered</tokenMessage>
    <type>VI</type>
    <bin>402410</bin>
  </tokenResponse>
</authorizationResponse>

```

Example: Online Authorization Response with Account Updater Token Info

In this example, the <accountUpdater> contains both original and new card information as well as the <extendedCardResponse> element. This signifies that the card number changed from the original to the new and (from the extended response code) that the issuer is reporting that the new account is closed. Although the Authorization was approved, this information allows you to make an informed decision about how to proceed with the sale.

```
<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <authorizationResponse id="834262" reportGroup="ABC Division">
    <littleTxnId>969506</littleTxnId>
    <orderId>65347567</orderId>
    <response>000</response>
    <responseTime>2011-07-25T15:13:43</responseTime>
    <postDate>2011-07-25</postDate>
    <message>Approved</message>
    <authCode>123457</authCode>
    <accountUpdater>
      <originalCardTokenInfo>
        <littleToken>1111100100240005</littleToken>
        <type>VI</type>
        <expDate>1112</expDate>
        <bin>400555</bin>
      </originalCardTokenInfo>
      <newCardTokenInfo>
        <littleToken>1111100100250017</littleToken>
        <type>VI</type>
        <expDate>1114</expDate>
        <bin>400555</bin>
      </newCardTokenInfo>
      <extendedCardResponse>
        <code>501</code>
        <message>The account was closed</message>
      </extendedCardResponse>
    </accountUpdater>
    <fraudResult>
      <avsResult>00</avsResult>
      <cardValidationResult>N</cardValidationResult>
      <authenticationResult>2</authenticationResult>
    </fraudResult>
  </authorizationResponse>
</littleOnlineResponse>
```

3.3.2 Authorization Reversal Transactions

The Authorization Reversal transaction enables you to remove the hold on any funds being held by an Authorization. The original Authorization transaction must have been processed within the Little system. For information on how to use the Authorization Reversal transaction, see [Notes on the Use of Authorization Reversal Transactions](#) on page 36. Also, if you use Little's Recycling Engine, you can use the `authReversal` transaction to halt the recycling of an authorization transaction.

3.3.2.1 Authorization Reversal Requests

You must structure an Authorization Reversal request as shown in the following examples. The structure of the request is identical for either an Online or a Batch submission.

```
<authReversal id="Authorization Id" reportGroup="UI Report Group"
customerId="Customer Id">

  <littleTxnId>Little & Co. Transaction Id</littleTxnId>

  <amount>Authorization Amount to Reverse</amount>

  <actionReason>SUSPECT_FRAUD</actionReason>

</authReversal>
```

Example: Batch Authorization Reversal Request

The following example contains three Authorization Reversal Requests.

```
<littleRequest version="8.10" xmlns="http://www.little.com/schema"
numBatchRequests="1">
  <authentication>
    <user>PHXMLTEST</user>
    <password>password</password>
  </authentication>
  <batchRequest numAuthReversals="3" authReversalAmount="3005"
merchantId="000057">
    <authReversal id="ID" customerId="customerId" reportGroup="000057">
      <littleTxnId>12345678</littleTxnId>
      <amount>1002</amount>
    </authReversal>
    <authReversal id="ID" customerId="customerId" reportGroup="000057">
      <littleTxnId>81234567</littleTxnId>
      <amount>1003</amount>
    </authReversal>
    <authReversal id="ID" customerId="customerId" reportGroup="000057">
      <littleTxnId>78123456</littleTxnId>
      <amount>1000</amount>
    </authReversal>
```



```

    </batchRequest>
</littleRequest>

```

Example: Online Authorization Reversal Request

```

<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <authReversal id="12345" customerId="Customer Id" reportGroup="Auth
  Reversals">
    <littleTxnId>12345678</littleTxnId>
    <amount>2999</amount>
  </authReversal>
</littleOnlineRequest>

```

3.3.2.2 Authorization Reversal Responses

The basic structure of an Authorization Reversal response is identical for Batch and Online responses.

```

<authReversalResponse id="Authorization Id" reportGroup="UI Report
Group" customerId="Customer Id">
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <message>Response Message</message>
</authReversalResponse>

```

Example: Batch Authorization Reversal Response

The following example shows a Batch Response containing three Authorization Reversal responses.

```

<littleResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="ValidFormat" littleSessionId="21500040809">
  <batchResponse littleBatchId="21500040908" merchantId="000902">
    <authReversalResponse id="ID" reportGroup="core" customerId="Rev1">
      <littleTxnId>21200000002700</littleTxnId>
      <orderId>diAuth2</orderId>
    </authReversalResponse>
  </batchResponse>
</littleResponse>

```

```

    <response>000</response>
    <responseTime>2011-10-14T13:15:43</responseTime>
    <message>Approved</message>
  </authReversalResponse>
  <authReversalResponse id="ID" reportGroup="core" customerId="viRev2">
    <littleTxnId>21200000002809</littleTxnId>
    <orderId>visaAuth2</orderId>
    <response>000</response>
    <responseTime>2011-10-14T13:15:43</responseTime>
    <message>Approved</message>
  </authReversalResponse>
  <authReversalResponse id="ID" reportGroup="core" customerId="mcRev3">
    <littleTxnId>21200000002908</littleTxnId>
    <orderId>mcAuth2</orderId>
    <response>000</response>
    <responseTime>2011-10-14T13:15:43</responseTime>
    <message>Approved</message>
  </authReversalResponse>
</batchResponse>
</littleResponse>

```

Example: Online Authorization Reversal Response

```

<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <authReversalResponse id="12345" customerId="Customer Id"
    reportGroup="Auth Reversals">
    <littleTxnId>12345678</littleTxnId>
    <orderId>abc123</orderId>
    <response>000</response>
    <responseTime>2011-08-30T13:15:43</responseTime>
    <message>Approved</message>
  </authReversalResponse>
</littleOnlineResponse>

```

3.3.3 Capture Transactions

The Capture transaction transfers funds from the customer to the merchant. The Capture references the associated Authorization by means of the `littleTxnId` element returned in the Authorization response.

You send a Capture after the order has been fulfilled. In some cases, it is not possible to fulfill a customer's entire order in one shipment (for example, if some items are backordered, or some shipped from an off-site DCS). In this situation, you can send a Partial Capture transaction by setting the `partial` attribute to **true**. A Partial Capture contains only the data relevant to the items that were actually shipped, enabling you to settle the funds related to those items.

3.3.3.1 Capture Request

You must structure a Capture request as shown in the following examples. The structure of the request is identical for either an Online or a Batch submission.

```
<capture id="Capture Id" reportGroup="UI Report Group" customerId="Customer Id"
partial="false">
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>
  <amount>Authorization Amount</amount>
  <enhancedData>
  <processingInstructions>
  <paypalOrderComplete>Set to true for final Capture</paypalOrderComplete>
</capture>
```

Example: Batch Capture Request - Full Capture

The following Capture example is for a full capture. Although the `<capture>` element includes an `<amount>` child, it is not required for a full Capture. If you omit the `<amount>` child element, the capture amount defaults to the full amount from the associated Authorization.

```
<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="01234567" numAuths="0" authAmount="0" numCaptures="1"
captureAmount="55814" numCredits="0" creditAmount="0" numSales="0"
saleAmount="0" merchantId="100">
    <capture id="AX54325432" reportGroup="RG12" partial="false">
      <littleTxnId>84568457</littleTxnId>
      <amount>55814</amount>
      <enhancedData>
```

```
<customerReference>PO12346</customerReference>
<salesTax>1500</salesTax>
<taxExempt>false</taxExempt>
<discountAmount>0</discountAmount>
<shippingAmount>3714</shippingAmount>
<dutyAmount>0</dutyAmount>
<shipFromPostalCode>01851</shipFromPostalCode>
<destinationPostalCode>01851</destinationPostalCode>
<destinationCountryCode>USA</destinationCountryCode>
<invoiceReferenceNumber>123456</invoiceReferenceNumber>
<orderDate>2011-09-14</orderDate>
<detailTax>
  <taxIncludedInTotal>true</taxIncludedInTotal>
  <taxAmount>500</taxAmount>
  <taxRate>0.01667</taxRate>
  <taxTypeIdentifier>00</taxTypeIdentifier>
  <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
</detailTax>
<lineItemData>
  <itemSequenceNumber>1</itemSequenceNumber>
  <itemDescription>table</itemDescription>
  <productCode>TB123</productCode>
  <quantity>1</quantity>
  <unitOfMeasure>EACH</unitOfMeasure>
  <taxAmount>1500</taxAmount>
  <lineItemTotal>30000</lineItemTotal>
  <lineItemTotalWithTax>31500</lineItemTotalWithTax>
  <itemDiscountAmount>0</itemDiscountAmount>
  <commodityCode>301</commodityCode>
  <unitCost>300.00</unitCost>
  <detailTax>
    <taxIncludedInTotal>true</taxIncludedInTotal>
    <taxAmount>500</taxAmount>
    <taxRate>0.01667</taxRate>
    <taxTypeIdentifier>03</taxTypeIdentifier>
    <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
  </detailTax>
</lineItemData>
<lineItemData>
  <itemSequenceNumber>2</itemSequenceNumber>
  <itemDescription>chair</itemDescription>
  <productCode>CH123</productCode>
  <quantity>1</quantity>
```

```

        <unitOfMeasure>EACH</unitOfMeasure>
        <lineItemTotal>20000</lineItemTotal>
        <itemDiscountAmount>0</itemDiscountAmount>
        <commodityCode>301</commodityCode>
        <unitCost>200.00</unitCost>
    </lineItemData>
</enhancedData>
</capture>
</batchRequest>
</littleRequest>

```

Example: Batch Capture Request - Partial Capture

A partial Capture has the `partial` attribute set to **true** and must include an `amount` child element.

```

<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="01234567" numAuths="0" authAmount="0" numCaptures="1"
    captureAmount="45814" numCredits="0" creditAmount="0" numSales="0"
    saleAmount="0" merchantId="100">
    <capture id="AX54325432" reportGroup="RG12" partial="true">
      <littleTxnId>84568457</littleTxnId>
      <amount>45814</amount>
      <enhancedData>
        <customerReference>PO12346</customerReference>
        <salesTax>2100</salesTax>
        <taxExempt>false</taxExempt>
        <discountAmount>0</discountAmount>
        <shippingAmount>3714</shippingAmount>
        <dutyAmount>0</dutyAmount>
        <shipFromPostalCode>01851</shipFromPostalCode>
        <destinationPostalCode>01851</destinationPostalCode>
        <destinationCountryCode>USA</destinationCountryCode>
        <invoiceReferenceNumber>123456</invoiceReferenceNumber>
        <orderDate>2011-09-14</orderDate>
        <detailTax>
          <taxIncludedInTotal>true</taxIncludedInTotal>
          <taxAmount>500</taxAmount>
          <taxRate>0.01667</taxRate>
        </detailTax>
      </enhancedData>
    </capture>
  </batchRequest>
</littleRequest>

```

```
<taxTypeIdentifier>00</taxTypeIdentifier>
<cardAcceptorTaxId>011234567</cardAcceptorTaxId>
</detailTax>
<lineItemData>
  <itemSequenceNumber>1</itemSequenceNumber>
  <itemDescription>table</itemDescription>
  <productCode>TB123</productCode>
  <quantity>1</quantity>
  <unitOfMeasure>EACH</unitOfMeasure>
  <taxAmount>1500</taxAmount>
  <lineItemTotal>30000</lineItemTotal>
  <lineItemTotalWithTax>31500</lineItemTotalWithTax>
  <itemDiscountAmount>0</itemDiscountAmount>
  <commodityCode>301</commodityCode>
  <unitCost>300.00</unitCost>
  <detailTax>
    <taxIncludedInTotal>true</taxIncludedInTotal>
    <taxAmount>500</taxAmount>
    <taxRate>0.01667</taxRate>
    <taxTypeIdentifier>03</taxTypeIdentifier>
    <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
  </detailTax>
</lineItemData>
<lineItemData>
  <itemSequenceNumber>2</itemSequenceNumber>
  <itemDescription>chair</itemDescription>
  <productCode>CH123</productCode>
  <quantity>1</quantity>
  <unitOfMeasure>EACH</unitOfMeasure>
  <lineItemTotal>20000</lineItemTotal>
  <itemDiscountAmount>0</itemDiscountAmount>
  <commodityCode>301</commodityCode>
  <unitCost>200.00</unitCost>
</lineItemData>
</enhancedData>
</capture>
</batchRequest>
</littleRequest>
```

Example: Online Capture Request - Full Capture

The following Capture example is for a full capture. Although the <capture> element includes an <amount> child, it is not required for a full Capture. If you omit the <amount> child element, the capture amount defaults to the full amount from the associated Authorization.

```
<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>password</password>
  </authentication>
  <capture id="2" reportGroup="ABC Division" customerId="038945"
    partial="false">
    <littleTxnId>13254123434</littleTxnId>
    <enhancedData>
      <customerReference>P012345</customerReference>
      <salesTax>125</salesTax>
      <taxExempt>false</taxExempt>
      <discountAmount>0</discountAmount>
      <shippingAmount>495</shippingAmount>
      <dutyAmount>0</dutyAmount>
      <shipFromPostalCode>01851</shipFromPostalCode>
      <destinationPostalCode>01851</destinationPostalCode>
      <destinationCountryCode>USA</destinationCountryCode>
      <invoiceReferenceNumber>123456</invoiceReferenceNumber>
      <orderDate>2011-08-14</orderDate>
      <detailTax>
        <taxIncludedInTotal>true</taxIncludedInTotal>
        <taxAmount>55</taxAmount>
        <taxRate>0.0059</taxRate>
        <taxTypeIdentifier>00</taxTypeIdentifier>
        <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
      </detailTax>
    <lineItemData>
      <itemSequenceNumber>1</itemSequenceNumber>
      <itemDescription>chair</itemDescription>
      <productCode>CH123</productCode>
      <quantity>1</quantity>
      <unitOfMeasure>EACH</unitOfMeasure>
      <taxAmount>125</taxAmount>
      <lineItemTotal>9380</lineItemTotal>
      <lineItemTotalWithTax>9505</lineItemTotalWithTax>
      <itemDiscountAmount>0</itemDiscountAmount>
    </lineItemData>
  </capture>
</littleOnlineRequest>
```

```

    <commodityCode>300</commodityCode>
    <unitCost>93.80</unitCost>
    <detailTax>
      <taxIncludedInTotal>true</taxIncludedInTotal>
      <taxAmount>55</taxAmount>
      <taxRate>0.0059</taxRate>
      <taxTypeIdentifier>03</taxTypeIdentifier>
      <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
    </detailTax>
  </lineItemData>
  <lineItemData>
    <itemSequenceNumber>2</itemSequenceNumber>
    <itemDescription>table</itemDescription>
    <productCode>TB123</productCode>
    <quantity>1</quantity>
    <unitOfMeasure>EACH</unitOfMeasure>
    <lineItemTotal>30000</lineItemTotal>
    <itemDiscountAmount>0</itemDiscountAmount>
    <commodityCode>300</commodityCode>
    <unitCost>300.00</unitCost>
  </lineItemData>
</enhancedData>
</capture>
</littleOnlineRequest>

```

Example: Online Capture Request - Partial Capture

A partial Capture has the **partial** attribute set to **true** and must include an `<amount>` child element.

```

<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>password</password>
  </authentication>
  <capture id="2" reportGroup="ABC Division" customerId="038945"
    partial="true">
    <littleTxnId>13254123434</littleTxnId>
    <amount>100</amount>
    <enhancedData>
      <customerReference>P012345</customerReference>
      <salesTax>125</salesTax>
    </enhancedData>
  </capture>
</littleOnlineRequest>

```



```
<taxExempt>>false</taxExempt>
<discountAmount>0</discountAmount>
<shippingAmount>495</shippingAmount>
<dutyAmount>0</dutyAmount>
<shipFromPostalCode>01851</shipFromPostalCode>
<destinationPostalCode>01851</destinationPostalCode>
<destinationCountryCode>USA</destinationCountryCode>
<invoiceReferenceNumber>123456</invoiceReferenceNumber>
<orderDate>2011-08-14</orderDate>
<detailTax>
  <taxIncludedInTotal>true</taxIncludedInTotal>
  <taxAmount>55</taxAmount>
  <taxRate>0.0059</taxRate>
  <taxTypeIdentifier>00</taxTypeIdentifier>
  <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
</detailTax>
<lineItemData>
  <itemSequenceNumber>1</itemSequenceNumber>
  <itemDescription>chair</itemDescription>
  <productCode>CH123</productCode>
  <quantity>1</quantity>
  <unitOfMeasure>EACH</unitOfMeasure>
  <taxAmount>125</taxAmount>
  <lineItemTotal>9380</lineItemTotal>
  <lineItemTotalWithTax>9505</lineItemTotalWithTax>
  <itemDiscountAmount>0</itemDiscountAmount>
  <commodityCode>300</commodityCode>
  <unitCost>93.80</unitCost>
  <detailTax>
    <taxIncludedInTotal>true</taxIncludedInTotal>
    <taxAmount>55</taxAmount>
    <taxRate>0.0059</taxRate>
    <taxTypeIdentifier>03</taxTypeIdentifier>
    <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
  </detailTax>
</lineItemData>
<lineItemData>
  <itemSequenceNumber>2</itemSequenceNumber>
  <itemDescription>table</itemDescription>
  <productCode>TB123</productCode>
  <quantity>1</quantity>
  <unitOfMeasure>EACH</unitOfMeasure>
  <lineItemTotal>30000</lineItemTotal>
```

```

        <itemDiscountAmount>0</itemDiscountAmount>
        <commodityCode>300</commodityCode>
        <unitCost>300.00</unitCost>
    </lineItemData>
</enhancedData>
</capture>
</littleOnlineRequest>

```

3.3.3.2 Capture Response

A Capture response has the following structure. The response message is identical for Online and Batch transactions except Batch always includes the `<orderId>` element, while Online includes the `<postDate>` element and may include a `duplicate` attribute.

```

<captureResponse id="Authorization Id" duplicate="true or false" reportGroup="UI
Report Group" customerId="Customer Id">
    <littleTxnId>Little & Co. Transaction Id</littleTxnId>
    <orderId>Order Id</orderId> (Batch Only)
    <response>Response Code</response>
    <responseTime>Date and Time in GMT</responseTime>
    <postDate>Date Of Posting</postDate> (Online Only)
    <message>Response Message</message>
    <accountUpdater>
</captureResponse>

```

Example: Batch Capture Response

```

<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
  littleSessionId="987654321" response="0" message="Valid Format">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <captureResponse id="AX54321678" reportGroup="RG27">
      <littleTxnId>84568456</littleTxnId>
      <orderId>12z58743y1</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>message</message>
    </captureResponse>
    <captureResponse id="AX54325432" reportGroup="RG12">
      <littleTxnId>84568457</littleTxnId>
      <orderId>12z58743y7</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
    </captureResponse>
  </batchResponse>
</littleResponse>

```

```

        <message>message</message>
    </captureResponse>
</batchResponse>
</littleResponse>

```

Example: Online Capture Response

NOTE: If the request is a duplicate (see [Online Duplicate Checking](#) on page 7), the response includes the duplicate attribute set to true and the entire original response.

```

<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <captureResponse id="2" reportGroup="ABC Division" customerId="038945">
        <littleTxnId>1100030204</littleTxnId>
        <response>000</response>
        <responseTime>2011-07-11T14:48:48</responseTime>
        <postDate>2011-07-11</postDate>
        <message>Approved</message>
    </captureResponse>
</littleOnlineResponse>

```

3.3.4 Capture Given Auth Transactions

You typically use a Capture Given Auth transaction when the associated Authorization occurred outside of the Little & Co. system (for example, if you received a telephone Authorization). Another possible use for a Capture Given Auth transaction is if the Authorization transaction occurred within the Little system, but the `<littleTxnId>` is unknown by the submitting party (for example, if the Auth was submitted by a merchant, but a fulfiller submits a Capture Given Auth).

3.3.4.1 Capture Given Auth Request

You must specify the Capture Given Auth request as follows. The structure of the request is identical for either an Online or a Batch submission.

```

<captureGivenAuth id="Capture Given Auth Id" reportGroup="UI Report Group"
    customerId="Customer Id">
    <orderId>Order Id</orderId>
    <authInformation>
    <amount>Authorization Amount</amount>

```

```

    <orderSource>Order Entry Source</orderSource>
    <billToAddress>
    <shipToAddress>
    [ <card> | <token> | <paypage> ]
    <customBilling>
    <taxType>payment or fee</taxType>
    <billMeLaterRequest>
    <enhancedData>
    <processingInstructions>
    <pos>
    <amexAggregatorData>
    <merchantData>
  </captureGivenAuth>

```

Example: Batch Capture Given Auth Request

The following example shows a single Capture Given Auth request. The example uses the <card> child element, but a tokenized merchant could use the <token> child element in its place.

```

<littleRequest version="8.10" xmlns="http://www.little.com/schema"
  numBatchRequests="1">
  <authentication>
    <user>XMLTEST</user>
    <password>password</password>
  </authentication>
  <batchRequest id="batchId" numCaptureGivenAuths="1"
    captureGivenAuthAmount="10000" merchantId="100">
    <captureGivenAuth id="AX54321678" reportGroup="RG27">
      <orderId>orderId</orderId>
      <authInformation>
        <authDate>2011-09-21</authDate>
        <authCode>123456</authCode>
      </authInformation>
      <amount>10000</amount>
      <orderSource>ecommerce</orderSource>
      <card>
        <type>VI</type>
        <number>4005550000081019</number>
        <expDate>0910</expDate>
      </card>
      <enhancedData>
    </captureGivenAuth>
  </batchRequest>
</littleRequest>

```

```

    <customerReference>P012345</customerReference>
    <salesTax>125</salesTax>
    <taxExempt>>false</taxExempt>
    <discountAmount>0</discountAmount>
    <shippingAmount>495</shippingAmount>
    <dutyAmount>0</dutyAmount>
    <shipFromPostalCode>01851</shipFromPostalCode>
    <destinationPostalCode>01851</destinationPostalCode>
    <destinationCountryCode>USA</destinationCountryCode>
    <invoiceReferenceNumber>123456</invoiceReferenceNumber>
    <orderDate>2011-09-21</orderDate>
    <detailTax>
      <taxIncludedInTotal>>true</taxIncludedInTotal>
      <taxAmount>55</taxAmount>
      <taxRate>0.0055</taxRate>
      <taxTypeIdentifier>00</taxTypeIdentifier>
      <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
    </detailTax>
    <lineItemData>
      <itemSequenceNumber>1</itemSequenceNumber>
      <itemDescription>chair</itemDescription>
      <productCode>CH123</productCode>
      <quantity>1</quantity>
      <unitOfMeasure>EACH</unitOfMeasure>
      <taxAmount>125</taxAmount>
      <lineItemTotal>9380</lineItemTotal>
      <lineItemTotalWithTax>9505</lineItemTotalWithTax>
      <itemDiscountAmount>0</itemDiscountAmount>
      <commodityCode>300</commodityCode>
      <unitCost>93.80</unitCost>
    </lineItemData>
  </enhancedData>
</captureGivenAuth>
</batchRequest>
</littleRequest>

```

Example: Online Capture Given Auth Request

```

<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>XMLTESTV3</user>
    <password>password</password>

```

```
</authentication>
<captureGivenAuth id="AX54321678" reportGroup="RG27">
  <orderId>orderId</orderId>
  <authInformation>
    <authDate>2011-08-24</authDate>
    <authCode>123456</authCode>
  </authInformation>
  <amount>10000</amount>
  <orderSource>ecommerce</orderSource>
  <card>
    <type>VI</type>
    <number>4005550000081019</number>
    <expDate>0910</expDate>
  </card>
  <enhancedData>
    <customerReference>P012345</customerReference>
    <salesTax>125</salesTax>
    <deliveryType>TBD</deliveryType>
    <taxExempt>>false</taxExempt>
    <discountAmount>0</discountAmount>
    <shippingAmount>495</shippingAmount>
    <dutyAmount>0</dutyAmount>
    <shipFromPostalCode>01851</shipFromPostalCode>
    <destinationPostalCode>01851</destinationPostalCode>
    <destinationCountryCode>USA</destinationCountryCode>
    <invoiceReferenceNumber>123456</invoiceReferenceNumber>
    <orderDate>2011-08-14</orderDate>
    <detailTax>
      <taxIncludedInTotal>>true</taxIncludedInTotal>
      <taxAmount>55</taxAmount>
      <taxRate>0.0059</taxRate>
      <taxTypeIdentifier>00</taxTypeIdentifier>
      <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
    </detailTax>
    <lineItemData>
      <itemSequenceNumber>1</itemSequenceNumber>
      <itemDescription>chair</itemDescription>
      <productCode>CH123</productCode>
      <quantity>1</quantity>
      <unitOfMeasure>EACH</unitOfMeasure>
      <taxAmount>125</taxAmount>
      <lineItemTotal>9380</lineItemTotal>
      <lineItemTotalWithTax>9505</lineItemTotalWithTax>
```

```
        <itemDiscountAmount>0</itemDiscountAmount>
        <commodityCode>300</commodityCode>
        <unitCost>93.80</unitCost>
    </lineItemData>
</enhancedData>
</captureGivenAuth>
</littleOnlineRequest>
```

Example: Capture Given Auth Request using token Element

The example below uses the following token related elements (click name to jump to element definition): [token](#) and [littleToken](#).

```
<captureGivenAuth id="99999" customerId="444" reportGroup="RG1">
    <orderId>F12345</orderId>
    <authInformation>
        <authDate>2011-10-25</authDate>
        <authCode>500005</authCode>
    </authInformation>
    <amount>15000</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
        <name>John Doe</name>
        <addressLine1>10 Main Street</addressLine1>
        <city>San Jose</city>
        <state>CA</state>
        <zip>95032-1234</zip>
        <country>USA</country>
    </billToAddress>
    <token>
        <littleToken>1112000100010085</littleToken>
        <expDate>1112</expDate>
        <cardValidationNum>987</cardValidationNum>
    </token>
</captureGivenAuth>
```

3.3.4.2 Capture Given Auth Response

A Capture Given Auth response has the following structure. The response message is identical for Online and Batch transactions except Online includes the <postDate> element and may include a duplicate attribute.

```
<captureGivenAuthResponse id="Capture Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date of Posting</postDate> (Online Only)
  <message>Response Message</message>
  <tokenResponse> (for Tokenized merchants submitting card data)
</captureGivenAuthResponse>
```

Example: Batch Capture Given Auth Response

```
<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
  littleSessionId="987654321" response="0" message="Valid Format">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <captureGivenAuthResponse id="AX54325432" reportGroup="RG12">
      <littleTxnId>84568457</littleTxnId>
      <orderId>12z58743y7</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>message</message>
    </captureGivenAuthResponse>
  </batchResponse>
</littleResponse>
```


Example: Online Capture Given Auth Response

NOTE: If the request is a duplicate (see [Online Duplicate Checking](#) on page 7), the response includes the duplicate attribute set to true and the entire original response.

```
<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <captureGivenAuthResponse id="2" reportGroup="ABC Division"
    customerId="038945">
    <littleTxnId>1100030204</littleTxnId>
    <orderId>65347567</orderId>
    <response>000</response>
    <responseTime>2011-07-11T14:48:48</responseTime>
    <postDate>2011-07-11</postDate>
    <message>Approved</message>
  </captureGivenAuthResponse>
</littleOnlineResponse>
```

Example: Capture Given Auth Response for Tokenized Merchant Sending Card Data

```
<captureGivenAuthResponse id="99999" reportGroup="RG1" customerId="444">
  <littleTxnId>21200000022005</littleTxnId>
  <orderId>F12345</orderId>
  <response>000</response>
  <responseTime>2011-10-25T04:00:00</responseTime>
  <postDate>2011-10-26</postDate>
  <message>Approved</message>
  <tokenResponse>
    <littleToken>1111000100409510</littleToken>
    <tokenResponseCode>801</tokenResponseCode>
    <tokenMessage>Account number was successfully registered</tokenMessage>
    <type>VI</type>
    <bin>432610</bin>
  </tokenResponse>
</captureGivenAuthResponse>
```

3.3.5 Credit Transactions

The Credit transaction enables you to refund money to a customer, even if the original transaction occurred outside of the Litle & Co. system. You can submit refunds against any of the following payment transactions:

- [Capture Transactions](#)
- [Capture Given Auth Transactions](#)
- [Force Capture Transactions](#)
- [Sale Transactions](#)
- External Sale or Capture Transactions

NOTE: Although there are two different scenarios for Credit requests, there is only one scenario for the Credit response.

3.3.5.1 Credit Request for a Litle & Co. Processed Transaction

You must specify the Credit request for a Litle & Co. processed transaction as follows. The structure of the request is identical for either an Online or a Batch submission.

```
<credit id="Credit Id" reportGroup="UI Report Group" customerId="Customer Id">
  <litleTxnId>Litle & Co. Transaction Id</litleTxnId>
  <amount>Authorization Amount</amount>
  <customBilling>
  <enhancedData>
  <processingInstructions>
</credit>
```

Example: Batch Credit Request for a Litle & Co. Processed Transaction

To request a Credit against a sale settled by Litle & Co., you only need to specify the `<litleTxnId>` element. The application uses the `<litleTxnId>` to look-up the Capture referenced and obtain all the necessary information including the amount.

NOTE: Although it is not required, if you choose to include `<amount>` elements in your Credit transaction, you must include the total amount in the `creditAmount` attribute of the `<batchrequest>`. If you do not specify amounts, set the `creditAmount` attribute to 0.

```
<litleRequest version="8.10" xmlns="http://www.litle.com/schema" id="123"
  numBatchRequests="1">
```

```
<authentication>
  <user>userName</user>
  <password>password</password>
</authentication>
<batchRequest id="01234567" numAuths="0" authAmount="0" numCaptures="0"
captureAmount="0" numCredits="1" creditAmount="10000" numSales="0"
saleAmount="0" merchantId="100">
  <credit id="AX54321678" reportGroup="RG27">
    <littleTxnId>84568456</littleTxnId>
    <amount>10000</amount>
    <enhancedData>
      <customerReference>PO12345</customerReference>
      <salesTax>125</salesTax>
      <taxExempt>>false</taxExempt>
      <discountAmount>0</discountAmount>
      <shippingAmount>3017</shippingAmount>
      <dutyAmount>0</dutyAmount>
      <shipFromPostalCode>01851</shipFromPostalCode>
      <destinationPostalCode>01851</destinationPostalCode>
      <destinationCountryCode>USA</destinationCountryCode>
      <invoiceReferenceNumber>123456</invoiceReferenceNumber>
      <orderDate>2011-09-14</orderDate>
      <detailTax>
        <taxIncludedInTotal>true</taxIncludedInTotal>
        <taxAmount>55</taxAmount>
        <taxRate>0.0059</taxRate>
        <taxTypeIdentifier>00</taxTypeIdentifier>
        <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
      </detailTax>
    </enhancedData>
  </credit>
</batchRequest>
<lineItemData>
  <itemSequenceNumber>1</itemSequenceNumber>
  <itemDescription>chair</itemDescription>
  <productCode>CH123</productCode>
  <quantity>1</quantity>
  <unitOfMeasure>EACH</unitOfMeasure>
  <taxAmount>125</taxAmount>
  <lineItemTotal>9380</lineItemTotal>
  <lineItemTotalWithTax>9505</lineItemTotalWithTax>
  <itemDiscountAmount>0</itemDiscountAmount>
  <commodityCode>300</commodityCode>
  <unitCost>93.80</unitCost>
  <detailTax>
    <taxIncludedInTotal>true</taxIncludedInTotal>
```

```

        <taxAmount>55</taxAmount>
        <taxRate>0.0059</taxRate>
        <taxTypeIdentifier>03</taxTypeIdentifier>
        <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
    </detailTax>
</lineItemData>
</enhancedData>
</credit>
</batchRequest>
</littleRequest>

```

Example: Online Credit Request for a Little & Co. Processed Transaction

To request a Credit against a sale settled by Little & Co., you need only specify the `<littleTxnId>` element. The application uses the `<littleTxnId>` to look up the Capture referenced and obtain all the necessary information including the amount. The example below includes the optional `<customBilling>` and `<enhancedData>` elements.

```

<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>password</password>
  </authentication>
  <credit id="2" reportGroup="ABC Division" customerId="038945">
    <littleTxnId>13254123434</littleTxnId>
    <customBilling>
      <phone>8888888888</phone>
      <descriptor>descriptor</descriptor>
    </customBilling>
    <enhancedData>
      <customerReference>P012345</customerReference>
      <salesTax>125</salesTax>
      <taxExempt>>false</taxExempt>
      <discountAmount>0</discountAmount>
      <shippingAmount>495</shippingAmount>
      <dutyAmount>0</dutyAmount>
      <shipFromPostalCode>01851</shipFromPostalCode>
      <destinationPostalCode>01851</destinationPostalCode>
      <destinationCountryCode>USA</destinationCountryCode>
      <invoiceReferenceNumber>123456</invoiceReferenceNumber>
      <orderDate>2011-07-14</orderDate>
      <detailTax>

```

```
<taxIncludedInTotal>true</taxIncludedInTotal>
<taxAmount>55</taxAmount>
<taxRate>0.0059</taxRate>
<taxTypeIdentifier>00</taxTypeIdentifier>
<cardAcceptorTaxId>011234567</cardAcceptorTaxId>
</detailTax>
<lineItemData>
  <itemSequenceNumber>1</itemSequenceNumber>
  <itemDescription>chair</itemDescription>
  <productCode>CH123</productCode>
  <quantity>1</quantity>
  <unitOfMeasure>EACH</unitOfMeasure>
  <taxAmount>125</taxAmount>
  <lineItemTotal>9380</lineItemTotal>
  <lineItemTotalWithTax>9505</lineItemTotalWithTax>
  <itemDiscountAmount>0</itemDiscountAmount>
  <commodityCode>300</commodityCode>
  <unitCost>93.80</unitCost>
  <detailTax>
    <taxIncludedInTotal>true</taxIncludedInTotal>
    <taxAmount>55</taxAmount>
    <taxRate>0.0059</taxRate>
    <taxTypeIdentifier>03</taxTypeIdentifier>
    <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
  </detailTax>
</lineItemData>
<lineItemData>
  <itemSequenceNumber>2</itemSequenceNumber>
  <itemDescription>table</itemDescription>
  <productCode>TB123</productCode>
  <quantity>1</quantity>
  <unitOfMeasure>EACH</unitOfMeasure>
  <lineItemTotal>30000</lineItemTotal>
  <itemDiscountAmount>0</itemDiscountAmount>
  <commodityCode>300</commodityCode>
  <unitCost>300.00</unitCost>
</lineItemData>
</enhancedData>
</credit>
</littleOnlineRequest>
```

3.3.5.2 Credit Request for a Non-Litle & Co. Processed Transaction

You must specify the Credit request for a Non-Litle & Co. processed transaction as follows. The structure of the request is identical for either an Online or a Batch submission.

NOTE: Although the schema shows `<paypal>` as an optional element for a Credit against a non-Litle & Co. processed transaction, refunds of this type are not supported for PayPal. If you need to refund non-Litle & Co. processed transactions and have not maintained a temporary relationship with your former processor for this purpose, please ask your Litle Relationship Manager for alternative options.

```
<credit id="Credit Id" reportGroup="UI Report Group" customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Authorization Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
    [ <card> | <token> | <paypal> ]
  <customBilling>
  <taxType>payment or fee</taxType>
  <billMeLaterRequest>
  <enhancedData>
  <processingInstructions>
  <pos>
  <amexAggregatorData>
  <merchantData>
</credit>
```

Example: Batch Credit Request for a Non-Litle & Co. Processed Transaction

If the sale occurred outside of the Litle & Co. system, you must specify the following elements in your Credit request: `<orderId>`, `<amount>`, and `<card>`, or `<token>` (`<paypal>` not supported for this transaction type).

```
<litleRequest version="8.10" xmlns="http://www.litle.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="01234567" numCredits="1" creditAmount="10000">
```

```
merchantId="100">
  <credit id="AX54321678" reportGroup="RG27">
    <orderId>12z58743y1</orderId>
    <amount>10000</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
      <name>John Doe</name>
      <addressLine1>123 4th street</addressLine1>
      <addressLine2>Apt. 20</addressLine2>
      <addressLine3>second floor</addressLine3>
      <city>San Jose</city>
      <state>CA</state>
      <zip>95032</zip>
      <country>USA</country>
      <email>jdoe@isp.com</email>
      <phone>408-555-1212</phone>
    </billToAddress>
    <card>
      <type>MC</type>
      <number>5186005800001012</number>
      <expDate>1110</expDate>
    </card>
    <enhancedData>
      <customerReference>PO12345</customerReference>
      <salesTax>125</salesTax>
      <taxExempt>>false</taxExempt>
      <discountAmount>0</discountAmount>
      <shippingAmount>495</shippingAmount>
      <dutyAmount>0</dutyAmount>
      <shipFromPostalCode>01851</shipFromPostalCode>
      <destinationPostalCode>01851</destinationPostalCode>
      <destinationCountryCode>USA</destinationCountryCode>
      <invoiceReferenceNumber>123456</invoiceReferenceNumber>
      <orderDate>2011-09-14</orderDate>
      <detailTax>
        <taxIncludedInTotal>true</taxIncludedInTotal>
        <taxAmount>55</taxAmount>
        <taxRate>0.0059</taxRate>
        <taxTypeIdentifier>00</taxTypeIdentifier>
        <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
      </detailTax>
      <lineItemData>
        <itemSequenceNumber>1</itemSequenceNumber>
```

```

        <itemDescription>chair</itemDescription>
        <productCode>CH123</productCode>
        <quantity>1</quantity>
        <unitOfMeasure>EACH</unitOfMeasure>
        <taxAmount>125</taxAmount>
        <lineItemTotal>9380</lineItemTotal>
        <lineItemTotalWithTax>9505</lineItemTotalWithTax>
        <itemDiscountAmount>0</itemDiscountAmount>
        <commodityCode>300</commodityCode>
        <unitCost>93.80</unitCost>
    </lineItemData>
</enhancedData>
</credit>
</batchRequest>
</litleRequest>

```

Example: Online Credit Request for a Non-Little & Co. Processed Transaction

If the sale occurred outside of the Little & Co. system, you must specify the following elements in your Credit request: <orderId>, <amount>, and <card>, or <token> (<paypal> not supported for this transaction type).

```

<litleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>password</password>
  </authentication>
  <credit id="2" reportGroup="ABC Division" customerId="038945">
    <orderId>56789</orderId>
    <amount>10000</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
      <name>Mike J. Hammer</name>
      <addressLine1>Two Main Street</addressLine1>
      <addressLine2>Apartment 222</addressLine2>
      <addressLine3></addressLine3>
      <city>Riverside</city>
      <state>RI</state>
      <zip>02915</zip>
      <country>US</country>
      <email>mike@sample.com</email>
      <phone>5555555555</phone>
    </billToAddress>
  </credit>
</litleOnlineRequest>

```



```
</billToAddress>
<card>
  <type>VI</type>
  <number>4005550000081019</number>
  <expDate>0907</expDate>
</card>
<customBilling>
  <phone>5555555555</phone>
  <descriptor>descriptor</descriptor>
</customBilling>
<enhancedData>
  <customerReference>P012345</customerReference>
  <salesTax>125</salesTax>
  <taxExempt>false</taxExempt>
  <discountAmount>0</discountAmount>
  <shippingAmount>495</shippingAmount>
  <dutyAmount>0</dutyAmount>
  <shipFromPostalCode>01851</shipFromPostalCode>
  <destinationPostalCode>01851</destinationPostalCode>
  <destinationCountryCode>USA</destinationCountryCode>
  <invoiceReferenceNumber>123456</invoiceReferenceNumber>
  <orderDate>2011-08-14</orderDate>
  <detailTax>
    <taxIncludedInTotal>true</taxIncludedInTotal>
    <taxAmount>55</taxAmount>
    <taxRate>0.0059</taxRate>
    <taxTypeIdentifier>00</taxTypeIdentifier>
    <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
  </detailTax>
  <lineItemData>
    <itemSequenceNumber>1</itemSequenceNumber>
    <itemDescription>chair</itemDescription>
    <productCode>CH123</productCode>
    <quantity>1</quantity>
    <unitOfMeasure>EACH</unitOfMeasure>
    <taxAmount>125</taxAmount>
    <lineItemTotal>9380</lineItemTotal>
    <lineItemTotalWithTax>9505</lineItemTotalWithTax>
    <itemDiscountAmount>0</itemDiscountAmount>
    <commodityCode>300</commodityCode>
    <unitCost>93.80</unitCost>
  </lineItemData>
  <lineItemData>
```

```

    <itemSequenceNumber>2</itemSequenceNumber>
    <itemDescription>table</itemDescription>
    <productCode>TB123</productCode>
    <quantity>1</quantity>
    <unitOfMeasure>EACH</unitOfMeasure>
    <lineItemTotal>30000</lineItemTotal>
    <itemDiscountAmount>0</itemDiscountAmount>
    <commodityCode>300</commodityCode>
    <unitCost>300.00</unitCost>
  </lineItemData>
</enhancedData>
</credit>
</littleOnlineRequest>

```

3.3.5.3 Credit Response

The Credit response message is identical for Online and Batch transactions except Online includes the `postDate` element and may include a `duplicate` attribute.

```

<creditResponse id="Credit Id" reportGroup="UI Report Group"
customerId="Customer Id">

  <littleTxnId>Little & Co. Transaction Id</littleTxnId>

  <orderId>Order Id</orderId>

  <response>Response Code</response>

  <responseTime>Date and Time in GMT</responseTime>

  <postDate>Date of Posting</postDate> (Online Only)

  <message>Response Message</message>

  <tokenResponse> (for Tokenized merchants submitting card data)

</creditResponse>

```

Example: Credit Response

The example below illustrates a Batch Credit response. A response for an Online transaction uses a `littleOnlineResponse` element as the parent.

```

<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
  littleSessionId="987654321" response="0" message="Valid Format">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <creditResponse id="AX54325432" reportGroup="RG12">
      <littleTxnId>84568457</littleTxnId>
      <orderId>12z58743y7</orderId>
      <response>000</response>
    </creditResponse>
  </batchResponse>
</littleResponse>

```

```
<responseTime>2011-09-01T10:24:31</responseTime>  
<message>Approved</message>  
</creditResponse>  
</batchResponse>  
</littleResponse>
```

Example: Credit Response for a Tokenized Merchant Sending Card Data

When a tokenized merchant submits card data in the Credit request, the response includes the `tokenResponse` element. The example below is a response for an Online request (`<litleOnlineResponse>` element not shown), so it includes the `<postDate>` element.

```
<creditResponse id="99999" reportGroup="RG1" customerId="444">
  <litleTxnId>21200000473208</litleTxnId>
  <orderId>F12345</orderId>
  <response>000</response>
  <responseTime>2011-10-19T19:29:45</responseTime>
  <postDate>2011-10-19</postDate>
  <message>Approved</message>
  <tokenResponse>
    <litleToken>1111102200202001</litleToken>
    <tokenResponseCode>801</tokenResponseCode>
    <tokenMessage>Account number was successfully registered</tokenMessage>
    <type>VI</type>
    <bin>402410</bin>
  </tokenResponse>
</creditResponse>
```

3.3.6 eCheck Credit Transactions

The eCheck Credit transaction enables you to refund a previous eCheck Sale. Merchants can submit an eCheck Credit transaction for a sale regardless of whether the original transaction was settled by the Litle & Co. system, although the requests are structured differently. This section contains the following:

- [eCheck Credit Request Against a Litle & Co. Transaction](#)
- [eCheck Credit Request for a Non-Litle & Co. Processed Sale](#)
- [eCheck Credit Response](#)

NOTE: Although there are two different scenarios for eCheck Credit requests, the response message uses the same structure.

3.3.6.1 eCheck Credit Request Against a Litle & Co. Transaction

To request an eCheck Credit against an eCheck Sale that had been settled by Litle & Co. you only need to specify the `<litleTxnId>` element. When you specify this element, the application uses the `<litleTxnId>` to look up the referenced echeckSale transaction and obtain the necessary information. In this case, the `<amount>` element is optional, but should be included if the credit

amount is less than the captured amount. If you do not include the `<amount>` element, the system assumes the credit to be for the total amount of the referenced transaction.

When requesting a `echeckCredit` against an `echeckSale` that occurred within Litle & Co., specify the Credit request as follows:

```
<echeckCredit id="Credit Id" reportGroup="UI Report Group" customerId="Customer
Id">

  <litleTxnId>Litle & Co. Transaction Id</litleTxnId>

  <amount>Credit Amount</amount>

  <customBilling>

</echeckCredit>
```

Example: eCheck Credit Request

The eCheck Credit batch request shown below contains three `<echeckCredit>` elements. The first two use a Litle Transaction ID as a reference. The third, which you would use for a sale occurring outside of the Litle & Co. system, uses the `<orderId>`, `<amount>`, `<billToAddress>`, and `<echeck>` elements to provide the required information.

```
<litleRequest version="8.10" xmlns="http://www.litle.com/schema"
numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="xmlbat01" numEcheckCredit="3" echeckCreditAmount="12100"
merchantId="000053">
    <echeckCredit id="credit1" reportGroup="new53" customerId="53">
      <litleTxnId>4455667788</litleTxnId>
      <amount>1000</amount>
    </echeckCredit>
    <echeckCredit reportGroup="new53">
      <litleTxnId>4455667789</litleTxnId>
      <amount>1100</amount>
    </echeckCredit>
    <echeckCredit reportGroup="new53">
      <orderId>12z58743y1</orderId>
      <amount>10000</amount>
      <orderSource>ecommerce</orderSource>
      <billToAddress>
        <name>John Doe</name>
        <addressLine1>123 4th street</addressLine1>
        <addressLine2>Apt. 20</addressLine2>
        <addressLine3>second floor</addressLine3>
```

```

    <city>San Jose</city>
    <state>CA</state>
    <zip>95032</zip>
    <country>USA</country>
    <email>jdoe@isp.com</email>
    <phone>408-555-1212</phone>
  </billToAddress>
  <echeck>
    <accType>Checking</accType>
    <accNum>5186005800001012</accNum>
    <routingNum>000010101</routingNum>
    <checkNum>1104</checkNum>
  </echeck>
</echeckCredit>
</batchRequest>
</litleRequest>

```

3.3.6.2 eCheck Credit Request for a Non-Litle & Co. Processed Sale

If the original eCheck Sale transaction was not processed via Litle & Co. or the if the `<litleTxnId>` for the original eCheck Sale transaction is not available, specify eCheck Credit request as follows:

```

<echeckCredit id="eCheckCredit Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Credit Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
  <echeck> or <echeckToken>
  <customBilling>
  <merchantData>
</echeckCredit>

```

The third transaction shown in the [eCheck Credit Request](#) example on page 153 shows an example of a Credit request against a non-Litle & Co. processed transaction.

3.3.6.3 eCheck Credit Response

The eCheck Credit message is identical for either type of eCheck Credit request. The `<accountUpdater>` element is included only if you submit account information in the request transaction for which a NOC exists. In this case the system automatically updates the information sent to the ACH network and includes the change information in the response.

The eCheck Credit response has the following structure:

```
<echeckCreditResponse id="eCheck Credit Id" reportGroup="UI Report Group"
customerId="Customer Id">

  <littleTxnId>Little & Co. Transaction Id</littleTxnId>

  <orderId>Order Id</orderId>

  <response>Response Code</response>

  <responseTime>Date and Time in GMT</responseTime>

  <message>Response Message</message>

  <postDate>Date of Posting</postDate> (Online Only)

  <accountUpdater>Account Change Info</accountUpdater>

  <tokenResponse> (for Tokenized merchants submitting account data)

</echeckCreditResponse>
```

Example: eCheck Credit Response

```
<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <echeckCreditResponse id="AX54321678" reportGroup="RG27"
customerId="53">
      <littleTxnId>84568456</littleTxnId>
      <orderId>12z58743y1</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>Approved</message>
    </echeckCreditResponse>
  </batchResponse>
</littleResponse>
```

3.3.7 eCheck Redeposit Transactions

You use this transaction type to manually attempt redeposits of eChecks returned for either Insufficient Funds or Uncollected Funds. You can use this element in either Batch or Online transactions.

NOTE: Do not use this transaction type if you are enabled for the Auto Redeposit feature. If you are enabled for the Auto Redeposit feature, the system will reject any `echeckRedeposit` transaction you submit.

3.3.7.1 eCheck Redeposit Request

You must specify the eCheck Redeposit request using the following format:

```
<echeckRedeposit id="eCheck Redeposit Id" reportGroup="UI Report Group"
customerId="Customer Id">

  <littleTxnId>Little Transaction Id</littleTxnId>

  <echeck> or <echeckToken>

</echeckRedeposit>
```

NOTE: If you include the `echeck` element, the values submitted for `accType`, `accNum`, and `routingNum` children must match those submitted in the original `echeckSale` transaction.

Example: Online eCheck Redeposit Request

```
<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
merchantId="81603">
  <authentication>
    <user>User Name</user>
    <password>password</password>
  </authentication>
  <echeckRedeposit reportGroup="001603">
    <littleTxnId>345454444</littleTxnId>
    <echeck>
      <accType>Checking</accType>
      <accNum>1099999903</accNum>
      <routingNum>114567895</routingNum>
    </echeck>
  </echeckRedeposit>
</littleOnlineRequest>
```


Example: Batch eCheck Redeposit Request

```

<littleRequest version="8.10" xmlns="http://www.little.com/schema"
  numBatchRequests="1">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <batchRequest id="uniqueId" numEcheckRedeposit="4" merchantId="1603">
    <echeckRedeposit reportGroup="001603">
      <littleTxnId>3456456444</littleTxnId>
    </echeckRedeposit>
    <echeckRedeposit reportGroup="001603">
      <littleTxnId>3456456449</littleTxnId>
      <echeck>
        <accType>Checking</accType>
        <accNum>1099999903</accNum>
        <routingNum>114567895</routingNum>
      </echeck>
    </echeckRedeposit>
    <echeckRedeposit reportGroup="001603">
      <littleTxnId>3456557123</littleTxnId>
      <echeck>
        <accType>Savings</accType>
        <accNum>10999999444</accNum>
        <routingNum>114567895</routingNum>
      </echeck>
    </echeckRedeposit>
    <echeckRedeposit reportGroup="001603">
      <littleTxnId>123456789</littleTxnId>
    </echeckRedeposit>
  </batchRequest>
</littleRequest>

```

3.3.7.2 eCheck Redeposit Response

The eCheck Redeposit response indicates that Little & Co. has received your eCheck Redeposit request. This does not indicate when funds will be transferred. The `<accountUpdater>` element is included only if the account information submitted in the request transaction has changed (NOC exists). In this case the system automatically updates the information sent to the ACH network and includes the change information in the response.

The eCheck Sale response has the following structure:

```

<echeckRedepositResponse id="eCheckRedeposit Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <message>Response Message</message>
  <postDate>Date of Posting</postDate> (Online Only)
  <accountUpdater>Account Change Info</accountUpdater>
</echeckRedepositResponse>

```

Example: Batch eCheck Redeposit Response

```

<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <echeckRedepositResponse id="AX54321678" reportGroup="RG27"
customerId="53">
      <littleTxnId>84568456</littleTxnId>
      <response>000</response>
      <responseTime>2010-06-01T10:24:31</responseTime>
      <message>Approved</message>
    </echeckRedepositResponse>
    <echeckRedepositResponse id="AX54325432" reportGroup="RG12">
      <littleTxnId>84568457</littleTxnId>
      <response>000</response>
      <responseTime>2010-06-01T10:24:31</responseTime>
      <message>Approved</message>
      <accountUpdater>
        <originalAccountInfo>
          <accType>Checking</accType>
          <accNum>5186005800001012</accNum>
          <routingNum>000010101</routingNum>
        </originalAccountInfo>
        <newAccountInfo>
          <accType>Checking</accType>
          <accNum>5499576040500006</accNum>
          <routingNum>000010102</routingNum>
        </newAccountInfo>
      </accountUpdater>
    </echeckRedepositResponse>
  </batchResponse>
</littleResponse>

```

3.3.8 eCheck Sale Transactions

You use the eCheck Sale transaction to capture funds from a customer paying via electronic checks. It is the eCheck equivalent of a Capture transaction. Setting the `<verify>` element to **true** triggers an eCheck Verification operation prior to the capture. If the verification fails, the system does not process the capture operation.

NOTE: To perform a verification you must include the following optional children of the `billToAddress` element in your request: `firstName`, `lastName`, `companyName` (if `accType` = Corporate or Corp Savings), `address1` (address 2 and 3 if needed), `city`, `state`, `phone`.

3.3.8.1 eCheck Sale Request

You must specify the eCheck Sale request using the following format:

```
<echeckSale id="eCheckSale Id" reportGroup="UI Report Group"
customerId="Customer Id">

  <orderId>Order Id</orderId>

  <verify>true or false</verify>

  <amount>Authorization Amount</amount>

  <orderSource>Order Entry Source</orderSource>

  <billToAddress>

  <shipToAddress>

  <echeck> or <echeckToken>

  <customBilling>

  <merchantData>

</echeckSale>
```

Example: eCheck Sale Request

```
<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="654321" numEcheckSales="1" echeckSalesAmount="10000"
merchantId="100">
    <echeckSale id="AX54321678" reportGroup="RG27" customerId="53">
      <orderId>12z58743y1</orderId>
      <verify>true</verify>
```

```

<amount>10000</amount>
<orderSource>telephone</orderSource>
<billToAddress>
  <firstName>John</firstName>
  <lastName>Doe</lastName>
  <addressLine1>123 4th street</addressLine1>
  <addressLine2>Apt. 20</addressLine2>
  <addressLine3>second floor</addressLine3>
  <city>San Jose</city>
  <state>CA</state>
  <zip>95032</zip>
  <country>USA</country>
  <email>jdoe@isp.com</email>
  <phone>408-555-1212</phone>
</billToAddress>
<echeck>
  <accType>Checking</accType>
  <accNum>5186005800001012</accNum>
  <routingNum>000010101</routingNum>
  <checkNum>1104</checkNum>
</echeck>
</echeckSale>
</batchRequest>
</littleRequest>

```

3.3.8.2 eCheck Sale Response

The eCheck Sale response indicates that Little & Co. has received your eCheck Sale request. This does not indicate when funds will be transferred. The `<accountUpdater>` element is included only if the account information submitted in the request transaction has changed (NOC exists). In this case the system automatically updates the information sent to the ACH network and includes the change information in the response.

NOTE: The schema for `echeckSalesResponse` includes a `verificationCode` child element. This element is not used at this time.

The eCheck Sale response has the following structure:

```

<echeckSalesResponse id="eCheckSale Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <littleTxnId>Little Transaction Id</littleTxnId>

```

```

<orderId>Order Id</orderId>
<response>Response Code</response>
<responseTime>Date and Time in GMT</responseTime>
<message>Response Message</message>
<postDate>Date of Posting</postDate> (Online Only)
<accountUpdater>Account Change Info</accountUpdater>
<tokenResponse> (for Tokenized merchants submitting account data)
</echeckSaleResponse>

```

Example: eCheck Sale Response

The response example below includes the `accountUpdater` element, which indicates that there is a NOC against the account and provides the new account information. If the request used a token, the `accountUpdater` element would have children providing the original and new token information.

```

<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
  response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <echeckSalesResponse id="AX54321678" reportGroup="RG27" customerId="53">
      <littleTxnId>84568456</littleTxnId>
      <orderId>12z58743y1</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>Approved</message>
    </echeckSalesResponse>
    <echeckSalesResponse id="AX54325432" reportGroup="RG12">
      <littleTxnId>84568457</littleTxnId>
      <orderId>12z58743y7</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>Approved</message>
      <accountUpdater>
        <originalAccountInfo>
          <accType>Checking</accType>
          <accNum>5186005800001012</accNum>
          <routingNum>000010101</routingNum>
        </originalAccountInfo>
        <newAccountInfo>
          <accType>Checking</accType>
          <accNum>5499576040500006</accNum>
          <routingNum>000010102</routingNum>
        </newAccountInfo>
      </accountUpdater>
    </echeckSalesResponse>
  </batchResponse>
</littleResponse>

```

```

    </accountUpdater>
  </echeckSalesResponse>
</batchResponse>
</littleResponse>

```

3.3.9 eCheck Verification Transactions

You use an eCheck Verification transaction to initiate a comparison to a database containing information about checking accounts. The database may include information as to whether the account has been closed, as well as whether there is a history of undesirable behavior associated with the account/account holder.

NOTE: While eCheck Verification is a valuable tool that you can use to reduce possible fraud and loss, unlike a credit card authorization, it does not check for the availability of funds, nor does it place a hold on any funds.

3.3.9.1 eCheck Verification Request

You must specify the eCheck Verification request using the following format:

IMPORTANT: When requesting an eCheck Verification for a corporate account, you must include the `companyName` in addition to the `firstName`, `lastName`, `Address1` (2, 3), `city`, `state`, `zip` and `phone` in the `echeckVerification` request.

If you do not have the name of the check issuer, you can use a value of "unavailable" for the `firstName` and `lastName` elements.

```

<echeckVerification id="echeckVerification Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>
  <orderId>Order Id</orderId>
  <amount>Authorization Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
  <echeck> or <echeckToken>
</echeckVerification>

```

Example: eCheck Verification Request - Personal Checking

```

<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="654321" numEcheckVerification="1"
    echeckVerificationAmount="10000" merchantId="100">
    <echeckVerification id="AX54321678" reportGroup="RG27" customerId="53">
      <orderId>12z58743y1</orderId>
      <amount>10000</amount>
      <orderSource>telephone</orderSource>
      <billToAddress>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
        <addressLine1>123 4th street</addressLine1>
        <addressLine2>Apt. 20</addressLine2>
        <addressLine3>second floor</addressLine3>
        <city>San Jose</city>
        <state>CA</state>
        <zip>95032</zip>
        <country>USA</country>
        <email>jdoe@isp.com</email>
        <phone>408-555-1212</phone>
      </billToAddress>
      <echeck>
        <accType>Checking</accType>
        <accNum>5186005800001012</accNum>
        <routingNum>000010101</routingNum>
        <checkNum>1104</checkNum>
      </echeck>
    </echeckVerification>
  </batchRequest>
</littleRequest>

```

Example: eCheck Verification Request - Corporate Account

NOTE: If you do not have the name of the check issuer, you can use a value of "unavailable" for the `firstName` and `lastName` elements.

```

<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"

```

```

numBatchRequests="1">
<authentication>
  <user>userName</user>
  <password>password</password>
</authentication>
<batchRequest id="654321" numEcheckVerification="1"
echeckVerificationAmount="10000" merchantId="100">
  <echeckVerification id="AX54321678" reportGroup="RG27" customerId="53">
    <orderId>12z58743y1</orderId>
    <amount>10000</amount>
    <orderSource>telephone</orderSource>
    <billToAddress>
      <firstName>Paul</firstName>
      <lastName>Jones</lastName>
      <companyName>Widget Company</companyName>
      <addressLine1>123 4th street</addressLine1>
      <addressLine2>Apt. 20</addressLine2>
      <addressLine3>second floor</addressLine3>
      <city>San Jose</city>
      <state>CA</state>
      <zip>95032</zip>
      <country>USA</country>
      <email>pjones@isp.com</email>
      <phone>408-555-1212</phone>
    </billToAddress>
    <echeck>
      <accType>Corporate</accType>
      <accNum>5186005800001012</accNum>
      <routingNum>000010101</routingNum>
      <checkNum>1104</checkNum>
    </echeck>
  </echeckVerification>
</batchRequest>
</littleRequest>8.6

```

3.3.9.2 eCheck Verification Response

The <accountUpdater> element is included only if the account information submitted in the request transaction has changed (NOC exists). In this case the system automatically updates the information sent to the ACH network and includes the change information in the response.

The eCheck Verification response has the following structure:


```

<echeckVerificationResponse id="echeckVerification Id" reportGroup="UI Report
Group" customerId="Customer Id">

  <littleTxnId>Little & Co. Transaction Id</littleTxnId>

  <orderId>Order Id</orderId>

  <response>Response Code</response>

  <responseTime>Date and Time in GMT</responseTime>

  <message>Response Message</message>

  <postDate>Date of Posting</postDate> (Online Only)

  <tokenResponse> (for Tokenized merchants submitting account data)

  <accountUpdater>Account Change Info</accountUpdater>

</echeckSaleResponse>

```

Example: eCheck Verification Response

```

<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
  response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <echeckVerificationResponse id="AX54321678" reportGroup="RG27"
      customerId="53">
      <littleTxnId>84568456</littleTxnId>
      <orderId>12z58743y1</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>Approved</message>
    </echeckVerificationResponse>
    <echeckVerificationResponse id="AX54325432" reportGroup="RG12">
      <littleTxnId>84568457</littleTxnId>
      <orderId>12z58743y7</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>Approved</message>
      <accountUpdater>
        <originalAccountInfo>
          <accType>Checking</accType>
          <accNum>5186005800001012</accNum>
          <routingNum>000010101</routingNum>
        </originalAccountInfo>
        <newAccountInfo>
          <accType>Checking</accType>
          <accNum>5499576040500006</accNum>
          <routingNum>000010102</routingNum>
        </newAccountInfo>
      </accountUpdater>
    </echeckVerificationResponse>
  </batchResponse>
</littleResponse>

```

```

    </accountUpdater>
  </echeckVerificationResponse>
</batchResponse>
</littleResponse>

```

3.3.10 eCheck Void Transactions (Online Only)

You use an eCheck Void transaction to either halt automatic redeposit attempts of eChecks returned for either Insufficient Funds or Uncollected Funds, or cancel an eCheck Sale transaction, as long as the transaction has not yet settled. This also applies to merchant initiated redeposits. You can use this element only in Online transactions.

3.3.10.1 eCheck Void Request

The eCheck Void request references the `<littleTxnId>` of the previously approved transaction. You must structure an eCheck Void request as follows.

```

<echeckVoid id = "echeckVoid Id" reportGroup="UI Report Group">
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>
</echeckVoid>

```

Example: eCheck Void Request

```

<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="81601">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <echeckVoid id="101" reportGroup="001601">
    <littleTxnId>345454444</littleTxnId>
  </echeckVoid>
</littleOnlineRequest>

```

3.3.10.2 eCheck Void Response

The eCheck Void response message may also include a `duplicate` attribute. The eCheck Void response has the following structure.

```

<echeckVoidResponse id="eCheck Void Id" reportGroup="UI Report Group"
  numDeposits=>"1"
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>

```

```

<response>Response Code</response>
<responseTime>Date and Time in GMT</responseTime>
<postDate>Date of Posting</postDate>
<message>Response Message</message>
</echeckVoidResponse>

```

Example: Online eCheck Void Response

```

<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <echeckVoidResponse id="101" reportGroup="001601">
    <littleTxnId>21200000026600</littleTxnId>
    <response>000</response>
    <responseTime>2010-06-17T21:20:50</responseTime>
    <message>Approved</message>
    <postDate>2010-06-17</postDate>
  </echeckVoidResponse>
</littleOnlineResponse>

```

3.3.11 Force Capture Transactions

A Force Capture transaction is a Capture transaction used when you do not have a valid Authorization for the order, but have fulfilled the order and wish to transfer funds.

CAUTION: Merchants must be authorized by Little & Co. before submitting transactions of this type. In some instances, using a Force Capture transaction can lead to chargebacks and fines.

3.3.11.1 Force Capture Request

You must specify the Force Capture request as follows. The structure of the request is identical for either an Online or a Batch submission.

```

<forceCapture id="Id" reportGroup="UI Report Group" customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Force Capture Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
    [ <card | token> | <paypage> ]
  </billToAddress>

```

```

    <customBilling>
    <taxType>payment or fee</taxType>
    <enhancedData>
    <processingInstructions>
    <pos>
    <amexAggregatorData>
    <merchantData>
  </forceCapture>

```

Example: Batch Force Capture Request

```

<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="01234567" numForceCaptures="1"
    forceCaptureAmount="10000" merchantId="100">
    <forceCapture id="AX54321678" reportGroup="RG27" customerId="038945">
      <orderId>orderId</orderId>
      <amount>10000</amount>
      <orderSource>ecommerce</orderSource>
      <card>
        <type>VI</type>
        <number>4005550000081019</number>
        <expDate>0910</expDate>
      </card>
      <enhancedData>
        <customerReference>P012345</customerReference>
        <salesTax>125</salesTax>
        <taxExempt>false</taxExempt>
        <discountAmount>0</discountAmount>
        <shippingAmount>495</shippingAmount>
        <dutyAmount>0</dutyAmount>
        <shipFromPostalCode>01851</shipFromPostalCode>
        <destinationPostalCode>01851</destinationPostalCode>
        <destinationCountryCode>USA</destinationCountryCode>
        <invoiceReferenceNumber>123456</invoiceReferenceNumber>
        <orderDate>2011-09-14</orderDate>
        <detailTax>
          <taxIncludedInTotal>true</taxIncludedInTotal>

```

```

        <taxAmount>55</taxAmount>
        <taxRate>0.0059</taxRate>
        <taxTypeIdentifier>00</taxTypeIdentifier>
        <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
    </detailTax>
    <lineItemData>
        <itemSequenceNumber>1</itemSequenceNumber>
        <itemDescription>chair</itemDescription>
        <productCode>CH123</productCode>
        <quantity>1</quantity>
        <unitOfMeasure>EACH</unitOfMeasure>
        <taxAmount>125</taxAmount>
        <lineItemTotal>9380</lineItemTotal>
        <lineItemTotalWithTax>9505</lineItemTotalWithTax>
        <itemDiscountAmount>0</itemDiscountAmount>
        <commodityCode>300</commodityCode>
        <unitCost>93.80</unitCost>
        <detailTax>
            <taxIncludedInTotal>true</taxIncludedInTotal>
            <taxAmount>55</taxAmount>
            <taxRate>0.0059</taxRate>
            <taxTypeIdentifier>03</taxTypeIdentifier>
            <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
        </detailTax>
    </lineItemData>
</enhancedData>
</forceCapture>
</batchRequest>
</littleRequest>

```

Example: Online Force Capture Request

```

<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="123">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <forceCapture id="AX54321678" reportGroup="RG27" customerId="038945">
    <orderId>orderId</orderId>
    <amount>10000</amount>
    <orderSource>ecommerce</orderSource>
    <card>

```

```
<type>VI</type>
<number>4005550000081019</number>
<expDate>0907</expDate>
</card>
<enhancedData>
  <customerReference>P012345</customerReference>
  <salesTax>125</salesTax>
  <taxExempt>false</taxExempt>
  <discountAmount>0</discountAmount>
  <shippingAmount>495</shippingAmount>
  <dutyAmount>0</dutyAmount>
  <shipFromPostalCode>01851</shipFromPostalCode>
  <destinationPostalCode>01851</destinationPostalCode>
  <destinationCountryCode>USA</destinationCountryCode>
  <invoiceReferenceNumber>123456</invoiceReferenceNumber>
  <orderDate>2011-08-14</orderDate>
  <detailTax>
    <taxIncludedInTotal>true</taxIncludedInTotal>
    <taxAmount>55</taxAmount>
    <taxRate>0.0059</taxRate>
    <taxTypeIdentifier>00</taxTypeIdentifier>
    <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
  </detailTax>
  <lineItemData>
    <itemSequenceNumber>1</itemSequenceNumber>
    <itemDescription>chair</itemDescription>
    <productCode>CH123</productCode>
    <quantity>1</quantity>
    <unitOfMeasure>EACH</unitOfMeasure>
    <taxAmount>125</taxAmount>
    <lineItemTotal>9380</lineItemTotal>
    <lineItemTotalWithTax>9505</lineItemTotalWithTax>
    <itemDiscountAmount>0</itemDiscountAmount>
    <commodityCode>300</commodityCode>
    <unitCost>93.80</unitCost>
  </lineItemData>
</enhancedData>
</forceCapture>
</littleOnlineRequest>
```

3.3.11.2 Force Capture Response

The Force Capture response message is identical for Online and Batch transactions, except Online includes the <postDate> element and may include a duplicate attribute. The Force Capture response has the following structure:

```
<forceCaptureResponse id="Capture Id" reportGroup="UI Report Group"
customerId="Customer Id">

  <littleTxnId>Little & Co. Transaction Id</littleTxnId>

  <orderId>Order Id</orderId>

  <response>Response Code</response>

  <responseTime>Date and Time in GMT</responseTime>

  <postDate>Date of Posting</postDate> (Online Only)

  <message>Response Message</message>

  <tokenResponse> (for Tokenized merchants submitting card data)

  <accountUpdater>

</forceCaptureResponse>
```

Example: Batch Force Capture Response

```
<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
  littleSessionId="987654321" response="0" message="Valid Format">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <forceCaptureResponse id="AX54325432" reportGroup="RG12"
      customerId="038945">
      <littleTxnId>84568457</littleTxnId>
      <orderId>12z58743y7</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>Approved</message>
    </forceCaptureResponse>
  </batchResponse>
</littleResponse>
```

Example: Online Force Capture Response

If the Online request is a duplicate (see [Online Duplicate Checking](#) on page 7), the response includes the duplicate attribute set to true (not shown) and the entire original response.

```
<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <forceCaptureResponse id="2" reportGroup="ABC Division"
    customerId="038945">
    <littleTxnId>1100030204</littleTxnId>
```

```
<orderId>65347567</orderId>
<response>000</response>
<responseTime>2011-07-11T14:48:48</responseTime>
<postDate>2011-07-11</postDate>
<message>Approved</message>
</forceCaptureResponse>
</littleOnlineResponse>
```

Example: Force Capture Response for Tokenized Merchant Sending Card Data

A tokenized merchant that includes card information in the request receives a response message that includes the `token` element. The example below is an Online response.

```
<forceCaptureResponse id="99999" reportGroup="RG1" customerId="444">
  <littleTxnId>21200000039504</littleTxnId>
  <orderId>F12345</orderId>
  <response>000</response>
  <responseTime>2011-10-20T18:25:38</responseTime>
  <postDate>2011-10-20</postDate>
  <message>Approved</message>
  <tokenResponse>
    <littleToken>111310880008000</littleToken>
    <tokenResponseCode>801</tokenResponseCode>
    <tokenMessage>Account number was successfully registered</tokenMessage>
    <type>AX</type>
    <bin></bin>
  </tokenResponse>
</forceCaptureResponse>
```


3.3.12 Register Token Transactions

The Register Token transaction enables you to submit a credit card number, eCheck account number, or Pay Page Registration Id to Little & Co. and receive a token in return. While you can submit Register Token transactions at any time, typically, you would make use of this transactions when initially converting to the use of tokens. In this case you would submit large quantities of credit cards/eCheck account numbers in batch files and replace them in your database with the tokens returned.

NOTE: When initially tokenizing your customer database, Little & Co. recommends that you collect all distinct credit card numbers in your database and submit the information in one or more large batch files. When you receive the response file, parse the returned token information to your database, replacing the card numbers.

3.3.12.1 Register Token Request

You must specify the Register Token request as follows. The structure of the request is identical for either an Online or a Batch submission. The child elements used differ depending upon whether you are registering a credit card account, an eCheck account, or submitting a Pay Page Registration Id.

For credit cards:

```
<registerTokenRequest id="Id" reportGroup="UI Report Group">
  <orderId>Order Id</orderId>
  <accountNumber>Card Account Number</accountNumber>
</registerTokenRequest>
```

For eCheck accounts:

```
<registerTokenRequest id="Id" reportGroup="UI Report Group">
  <orderId>Order Id</orderId>
  <echeckForToken>
    <accNum>Account Number</accNum>
    <routingNum>Routing Number</routingNum>
  </echeckForToken>
</registerTokenRequest>
```

For Pay Page Registration Ids:

```
<registerTokenRequest id="Id" reportGroup="UI Report Group">
  <orderId>Order Id</orderId>
  <paypageRegistrationId>
```

```
</registerTokenRequest>
```

Example: Batch Register Token Request - Credit Card

```
<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="01234567" numTokenRegistrations="1">
    <registerTokenRequest id="99999" reportGroup="RG1">
      <orderId>F12345</orderId>
      <accountNumber>4005101001000002</accountNumber>
    </registerTokenRequest>
  </batchRequest>
</littleRequest>
```

Example: Batch Register Token Request - eCheck

```
<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="01234567" numTokenRegistrations="1">
    <registerTokenRequest id="99999" reportGroup="RG1">
      <orderId>F12345</orderId>
      <echeckForToken>
        <accNum>12345678901234567</accNum>
        <routingNum>000010101</routingNum>
      </echeckForToken>
    </registerTokenRequest>
  </batchRequest>
</littleRequest>
```

Example: Batch Register Token Request - paypageRegistrationId

```
<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
```

```

</authentication>
<batchRequest id="01234567" numTokenRegistrations="1">
  <registerTokenRequest id="99999" reportGroup="RG1">
    <orderId>F12345</orderId>
    <paypageRegistrationId>12345678901234567</paypageRegistrationId>
  </registerTokenRequest>
</batchRequest>
</littleRequest>

```

3.3.12.2 Register Token Response

There is no structural difference an Online and Batch response; however, some child elements change depending upon whether the token is for a credit card account or an eCheck account. The response will have one of the following structures.

Register Token response for Credit Cards

```

<registerTokenResponse id="99999" reportGroup="RG1">
  <littleTxnId>Little Transaction Number</littleTxnId>
  <orderId>Order Id</orderId>
  <littleToken>Little Token</littleToken>
  <bin>BIN</bin>
  <type>Method of Payment</type>
  <response>Response Code</response>
  <responseTime>Response Time</responseTime>
  <message>Response Message</message>
</registerTokenResponse>

```

Register Token response for eChecks

```

<registerTokenResponse id="99999" reportGroup="RG1">
  <littleTxnId>Little Transaction Number</littleTxnId>
  <orderId>Order Id</orderId>
  <littleToken>Little Token</littleToken>
  <type>Method of Payment</type>
  <eCheckAccountSuffix>Last 3 of Acct Number</eCheckAccountSuffix>
  <response>Response Code</response>
  <responseTime>Response Time</responseTime>
  <message>Response Message</message>

```

```
</registerTokenResponse>
```

Example: Batch Register Token Response - Credit Card

```
<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
  response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <registerTokenResponse id="99999" reportGroup="RG1">
      <littleTxnId>21122700</littleTxnId>
      <orderId>F12345</orderId>
      <littleToken>1111000100360002</littleToken>
      <bin>400510</bin>
      <type>VI</type>
      <response>801</response>
      <responseTime>2010-10-26T17:21:51</responseTime>
      <message>Account number was successfully registered</message>
    </registerTokenResponse>
  </batchResponse>
</littleResponse>
```

Example: Batch Register Token Request - eCheck

```
<littleResponse version="8.2" xmlns="http://www.little.com/schema" id="123"
  response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <registerTokenResponse id="99999" reportGroup="RG1">
      <littleTxnId>21122700</littleTxnId>
      <orderId>F12345</orderId>
      <littleToken>1111000100360002</littleToken>
      <type>VI</type>
      <eCheckAccountSuffix>511</eCheckAccountSuffix>
      <response>801</response>
      <responseTime>2010-10-26T17:21:51</responseTime>
      <message>Account number was successfully registered</message>
    </registerTokenResponse>
  </batchResponse>
</littleResponse>
```

3.3.13 RFR Transactions (Batch Only)

An RFR (Request For Response) transaction enables you to request a response file for a previously submitted Batch file. You make the request by submitting either the `littleSessionId` of the Batch, or in the case of a request for an Account Updater file, the `accountUpdateFileRequestData` element.

NOTE: The use of RFR transactions for Account Updater files apply only to the legacy Account Updater solution.

3.3.13.1 RFR Request

You must specify the RFR request as follows.

```
<RFRRequest>
  <littleSessionId | accNum>
</RFRRequest>
```

Example: RFR Request for Payment Transaction Batch

The following example shows an RFR request for the response, with 7766554321 as the value of the `<littleSessionId>` element.

```
<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="0">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <RFRRequest>
    <littleSessionId>7766554321</littleSessionId>
  </RFRRequest>
</littleRequest>
```

Example: RFR Request for an Account Updater File

```
<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="0">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <RFRRequest>
    <merchantId>100</merchantId>
```

```
<postDate>2010-01-15</postDate>
</RFRRequest>
</littleRequest>
```

3.3.13.2 RFR Response

When using an RFR request to obtain the response file for a payment transaction Batch file, the RFR response is exactly the same as the original session response associated with the `<littleSessionId>` you submitted in the RFR request. The session ID returned in the response will be the session ID of the original session.

When using an RFR request in an Account Updater scenario, you will receive either an Account Updater Completion response, if the file is ready, or an Account Updater RFR “not ready” response, as shown in the example below.

Example: Account Updater RFR “not ready” Response

```
<littleResponse version="8.10" xmlns="http://www.little.com/schema">
  <RFRResponse response="1" message="The account update file is not ready
  yet. Please try again later.">
  </RFRResponse>
</littleResponse>
```

3.3.14 Sale Transactions

The Sale transaction enables you to both authorize fund availability and deposit those funds by means of a single transaction. The Sale transaction is also known as a conditional deposit, because the deposit takes place only if the authorization succeeds. If the authorization is declined, the deposit will not be processed.

NOTE: If the authorization succeeds, the deposit is processed automatically, regardless of the AVS, CVV2, CVC2, or CID response.

3.3.14.1 Sale Request

You must specify the Sale request as follows. The structure of the request is identical for either an Online or a Batch submission.

```
<sale id="Sale Id" reportGroup="UI Report Group" customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Authorization Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <customerInfo>
  <billToAddress>
  <shipToAddress>
  [ <card> | <paypage> | <token> | <paypal> ]
  <billMeLaterRequest>
  <cardholderAuthentication>
  <customBilling>
  <taxType>payment or fee</taxType>
  <enhancedData>
  <processingInstructions>
  <pos>
  <paypalOrderComplete>Send true in the final Sale</paypalOrderComplete>
  <amexAggregatorData>
  <allowPartialAuth>
  <healthcareIIAS>
  <filtering>
  <merchantData>
  <recyclingRequest>
</sale>
```

Example: Batch Sale Request

```
<littleRequest version="8.10" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="01234567" numSales="1" saleAmount="12522"
    merchantId="100">
    <sale id="AX54321678" reportGroup="RG27">
      <orderId>12z58743y1</orderId>
      <amount>12522</amount>
      <orderSource>ecommerce</orderSource>
      <billToAddress>
        <name>John Doe</name>
        <addressLine1>123 4th street</addressLine1>
        <addressLine2>Apt. 20</addressLine2>
        <addressLine3>second floor</addressLine3>
        <city>San Jose</city>
        <state>CA</state>
        <zip>95032</zip>
        <country>USA</country>
        <email>jdoe@isp.com</email>
        <phone>408-555-1212</phone>
      </billToAddress>
      <card>
        <type>MC</type>
        <number>5186005800001012</number>
        <expDate>1110</expDate>
      </card>
    </sale>
  </batchRequest>
</littleRequest>
```


Example: Online Sale Request

NOTE: The example below includes an `<orderSource>` value of `3dsAuthenticated` and includes the `<cardholderAuthentication>` information. Use this `<orderSource>` value only if you are a 3DS merchant and authenticated the cardholder.

Also, the values for the `<authenticationValue>` and `<authenticationTransactionId>` elements in the example below have been truncated.

```
<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>password</password>
  </authentication>
  <sale id="1" reportGroup="ABC Division" customerId="038945">
    <orderId>5234234</orderId>
    <amount>40000</amount>
    <orderSource>3dsAuthenticated</orderSource>
    <billToAddress>
      <name>John Smith</name>
      <addressLine1>100 Main St</addressLine1>
      <addressLine2>100 Main St</addressLine2>
      <addressLine3>100 Main St</addressLine3>
      <city>Boston</city>
      <state>MA</state>
      <zip>12345</zip>
      <country>US</country>
      <email>jsmith@someaddress.com</email>
      <phone>555-123-4567</phone>
    </billToAddress>
    <card>
      <type>VI</type>
      <number>4005550000081019</number>
      <expDate>1210</expDate>
      <cardValidationNum>555</cardValidationNum>
    </card>
    <cardholderAuthentication>
      <authenticationValue>BwABBJQ1AgJDUCAAAAAA=</authenticationValue>
      <authenticationTransactionId>gMV75TAgk=</authenticationTransactionId>
    </cardholderAuthentication>
    <customBilling>
```

```
<phone>8888888888</phone>
<descriptor>bdi*Little&Co Test</descriptor>
</customBilling>
<enhancedData>
  <customerReference>P012345</customerReference>
  <salesTax>125</salesTax>
  <taxExempt>false</taxExempt>
  <discountAmount>0</discountAmount>
  <shippingAmount>495</shippingAmount>
  <dutyAmount>0</dutyAmount>
  <shipFromPostalCode>01851</shipFromPostalCode>
  <destinationPostalCode>01851</destinationPostalCode>
  <destinationCountryCode>USA</destinationCountryCode>
  <invoiceReferenceNumber>123456</invoiceReferenceNumber>
  <orderDate>2011-08-14</orderDate>
  <detailTax>
    <taxIncludedInTotal>true</taxIncludedInTotal>
    <taxAmount>55</taxAmount>
    <taxRate>0.0059</taxRate>
    <taxTypeIdentifier>00</taxTypeIdentifier>
    <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
  </detailTax>
  <lineItemData>
    <itemSequenceNumber>1</itemSequenceNumber>
    <itemDescription>chair</itemDescription>
    <productCode>CH123</productCode>
    <quantity>1</quantity>
    <unitOfMeasure>EACH</unitOfMeasure>
    <taxAmount>125</taxAmount>
    <lineItemTotal>9380</lineItemTotal>
    <lineItemTotalWithTax>9505</lineItemTotalWithTax>
    <itemDiscountAmount>0</itemDiscountAmount>
    <commodityCode>300</commodityCode>
    <unitCost>93.80</unitCost>
    <detailTax>
      <taxIncludedInTotal>true</taxIncludedInTotal>
      <taxAmount>55</taxAmount>
      <taxRate>0.0059</taxRate>
      <taxTypeIdentifier>03</taxTypeIdentifier>
      <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
    </detailTax>
  </lineItemData>
  <lineItemData>
```

```

        <itemSequenceNumber>2</itemSequenceNumber>
        <itemDescription>table</itemDescription>
        <productCode>TB123</productCode>
        <quantity>1</quantity>
        <unitOfMeasure>EACH</unitOfMeasure>
        <lineItemTotal>30000</lineItemTotal>
        <itemDiscountAmount>0</itemDiscountAmount>
        <commodityCode>300</commodityCode>
        <unitCost>300.00</unitCost>
    </lineItemData>
</enhancedData>
</sale>
</littleOnlineRequest>

```

3.3.14.2 Sale Response

The Sale response message is identical for Online and Batch transactions except Online includes the `postDate` element and may include a `duplicate` attribute. The Sale response has the following structure:

```

<saleResponse id="Sale Id" reportGroup="UI Report Group" customerId="Customer
Id">
    <littleTxnId>Little & Co. Transaction Id</littleTxnId>
    <orderId>Order Id</orderId>
    <response>Response Code</response>
    <responseTime>Date and Time in GMT</responseTime>
    <postDate>Date of Posting</postDate> (Online Only)
    <message>Response Message</message>
    <authCode>Approval Code</authCode>
    <approvedAmount>Approved amount for partial Auth</approvedAmount>
    <accountInformation>
    <fraudResult>
    <billMeLaterResponseData>
    <tokenResponse> (for Tokenized merchants submitting card data)
    <enhancedAuthResponse>
    <accountUpdater>
    <recycling> (included for declined Auths if featrure is enabled)
</saleResponse>

```

Example: Batch Sale Response

```
<littleResponse version="8.10" xmlns="http://www.little.com/schema" id="123"
  response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <saleResponse id="AX54321678" reportGroup="RG27">
      <littleTxnId>84568456</littleTxnId>
      <orderId>12z58743y1</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>Approved</message>
      <authCode>123456</authCode>
      <fraudResult>
        <avsResult>00</avsResult>
      </fraudResult>
    </saleResponse>
    <saleResponse id="AX54325432" reportGroup="RG12">
      <littleTxnId>84568457</littleTxnId>
      <orderId>12z58743y7</orderId>
      <response>000</response>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>Approved</message>
      <authCode>123456</authCode>
      <fraudResult>
        <avsResult>00</avsResult>
        <authenticationResult>2</authenticationResult>
      </fraudResult>
    </saleResponse>
  </batchResponse>
</littleResponse>
```

Example: Online Sale Response

```
<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <saleResponse id="1" reportGroup="ABC Division" customerId="038945">
    <littleTxnId>1100030055</littleTxnId>
    <orderId>23423434</orderId>
    <response>000</response>
    <responseTime>2011-07-11T14:48:46</responseTime>
    <postDate>2011-07-11</postDate>
    <message>Approved</message>
    <authCode>123457</authCode>
    <fraudResult>
```

```

    <avsResult>01</avsResult>
    <cardValidationResult>U</cardValidationResult>
    <authenticationResult>2</authenticationResult>
  </fraudResult>
</saleResponse>
</littleOnlineResponse>

```

Example: Online Sale Response for Tokenized Merchant Sending Card Data

A tokenized merchant that includes card information in the request receives a response message that includes the `token` element. The example below is an Online response.

```

<saleResponse id="99999" reportGroup="RG1" customerId="444">
  <littleTxnId>21200000028606</littleTxnId>
  <orderId>F12345</orderId>
  <response>000</response>
  <responseTime>2011-10-26T17:30:00</responseTime>
  <postDate>2011-10-26</postDate>
  <message>Approved</message>
  <authCode>089510</authCode>
  <fraudResult>
    <avsResult>11</avsResult>
    <cardValidationResult>P</cardValidationResult>
  </fraudResult>
  <tokenResponse>
    <littleToken>1111000100329510</littleToken>
    <tokenResponseCode>801</tokenResponseCode>
    <tokenMessage>Account number was successfully registered</tokenMessage>
    <type>VI</type>
    <bin>432610</bin>
  </tokenResponse>
</saleResponse>

```

Example: Online Sale Response with Account Updater Info

```

<littleOnlineResponse version="8.10" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <saleResponse id="1" reportGroup="ABC Division" customerId="038945">
    <littleTxnId>1100030055</littleTxnId>
    <orderId>23423434</orderId>
    <response>000</response>
    <responseTime>2011-07-11T14:48:46</responseTime>
    <postDate>2011-07-11</postDate>
  </saleResponse>
</littleOnlineResponse>

```

```

<message>Approved</message>
<authCode>123457</authCode>
<accountUpdater>
  <originalCardInfo>
    <type>VI</type>
    <number>4234823492346234</number>
    <expDate>1112</expDate>
  </originalCardInfo>
  <newCardInfo>
    <type>VI</type>
    <number>4234823490005777</number>
    <expDate>1114</expDate>
  </newCardInfo>
</accountUpdater>
<fraudResult>
  <avsResult>01</avsResult>
  <cardValidationResult>U</cardValidationResult>
  <authenticationResult>2</authenticationResult>
</fraudResult>
</saleResponse>
</littleOnlineResponse>

```

3.3.15 Void Transactions (Online Only)

You use a Void transaction to cancel a transaction that occurred during the same business day. You can void Capture, Credit, and Sale transactions.

NOTE: Do not use Void transactions to void an Authorization. To remove an Authorization use an Authorization Reversal transaction (see [Authorization Reversal Transactions](#) on page 124.)

If you attempt to void a transaction after the cutoff time, the system returns a response code of **362** and the message, **Transaction Not Voided - Already Settled**. In this situation, you can cancel the original transaction by using its reverse transaction, as shown in [Table 3-2](#).

TABLE 3-2 Cancelling a Transaction That Cannot Be Voided

If you had originally sent this transaction...	Cancel it by using this transaction...
Capture Transactions	Credit Transactions

TABLE 3-2 Cancelling a Transaction That Cannot Be Voided

If you had originally sent this transaction...	Cancel it by using this transaction...
Sale Transactions	Credit Transactions
Credit Transactions	Sale Transactions
Force Capture Transactions	Credit Transactions

3.3.15.1 Void Request

The Void request references the `<littleTxnId>` of the previously approved transaction. You must structure a Void request as follows.

```
<void id = "Void Id" reportGroup="UI Report Group">
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>
  <processingInstructions>
</void>
```

Example: Online Void Request

```
<littleOnlineRequest version="8.10" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <void id="1" reportGroup="Void Division">
    <littleTxnId>345454444</littleTxnId>
  </void>
</littleOnlineRequest>
```

3.3.15.2 Void Response

The Void response message may also include a `duplicate` attribute. The Void response has the following structure.

```
<voidResponse id="Void Id" reportGroup="UI Report Group">
  <littleTxnId>Little & Co. Transaction Id</littleTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
```

```
<postDate>Date of Posting</postDate>
<message>Response Message</message>
</voidResponse>
```

Example: Online Void Response

If the request is a duplicate (see [Online Duplicate Checking](#) on page 7), the response includes the duplicate attribute set to true and the entire original response.

```
<littleOnlineResponse version="8.10"
xmlns="http://www.little.com/schema" response="0" message="Valid
Format">
  <voidResponse id="1" reportGroup="Void Division">
    <littleTxnId>1100026202</littleTxnId>
    <response>000</response>
    <responseTime>2011-07-16T19:43:38</responseTime>
    <postDate>2011-07-16</postDate>
    <message>Approved</message>
  </voidResponse>
</littleOnlineResponse>
```




LITLEXML ELEMENTS

This chapter provides definitions for the elements and attributes used in LitleXML. This information is intended to be used in combination with the various LitleXML schema files to assist you as you build the code necessary to submit transactions to Litle & Co. transaction processing systems. Each section defines a particular element, its relationship to other elements (parents and children), as well as any attributes associated with the element.

For additional information on the structure of Litle XML requests and responses using these elements, as well as XML examples, please refer to [Chapter 3, "LitleXML Transaction Examples"](#).

The XML elements defined in this chapter are listed alphabetically.

4.1 accNum

The `accNum` element is a required child of the `echeck`, `originalAccountInfo`, and `newAccountInfo` elements defining the account number of the eCheck account.

Type = String; minLength = 4; maxLength = 17

NOTE:	Although the schema does not specify a minimum length for the <code>accNum</code> element, the number must be greater than or equal to 4 characters for the transaction to succeed.
--------------	---

Parent Elements:

[echeck](#), [newAccountInfo](#), [originalAccountInfo](#)

Attributes:

None

Child Elements:

None

4.2 accountInformation

The `accountInformation` element is an optional child of the `authorizationResponse` and `saleResponse` elements. It contains two children that define the card type and account number.

Parent Elements:

[authorizationResponse](#), [saleResponse](#)

Attributes:

None

Child Elements:

Required: [type](#)

Optional: [number](#)

4.3 accountNumber

The `accountNumber` element is a required child of the `registerTokenRequest` element defining the account number for which you are requesting a token.

Type = String; **minLength** = 13; **maxLength** = 25

Parent Elements:

[registerTokenRequest](#)

Attributes:

None

Child Elements:

None

4.4 accountUpdate

The `accountUpdate` element is the parent element for all Account Updater request transactions. You can use this only with Batch transactions.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements: (all Required)

[orderId](#), [cardOrToken](#) (allows the substitution of either the `card` or `token` elements)

4.5 accountUpdateFileRequestData

The `accountUpdateFileRequestData` element is a child of the `RFRRequest` element, required when requesting the response file for an (legacy) Account Updater submission.

Parent Elements:

[RFRRequest](#)

Attributes:

None

Child Elements:

Required: [merchantId](#)

Optional: [postDay](#)

Example: accountUpdateFileRequestData Structure

```
<accountUpdateFileRequestData>
  <merchantId>Merchant ID</merchantId>
  <postDay>Post Date</postDay>
</accountUpdateFileRequestData>
```

4.6 accountUpdater

The `accountUpdater` element is an optional child of the `authorizationResponse`, `captureResponse`, `echeckSalesResponse`, `echeckCreditResponse`, `echeckVerificationResponse`, `forceCaptureResponse`, and `saleResponse` elements. This element is included in the response messages when the submitted account information has changed.

In the case of eCheck accounts, the system automatically updates the information sent to the ACH network and includes the original and updated information in the response. Similarly, if you use the Automatic Account Updater service (for credit cards), the system automatically repairs the card information sent to the card networks and depending upon the option you select, can return the info to you.

Parent Elements:

[authorizationResponse](#), [captureResponse](#), [forceCaptureResponse](#), [echeckCreditResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [saleResponse](#)

Attributes:

None

Child Elements:

Required: [extendedCardResponse](#), [newAccountInfo](#), [newCardInfo](#), [newCardTokenInfo](#), [newTokenInfo](#), [originalAccountInfo](#), [originalCardInfo](#), [originalCardTokenInfo](#), [originalTokenInfo](#)

Example: accountUpdater Structure - Credit Cards without extendedCardResponse

```
<accountUpdater>
  <originalCardInfo>
    <type>Card Type</type>
    <number>Old Account Number</number>
    <expDate>Old Expiration Date</expDate>
  </originalCardInfo>
  <newCardInfo>
    <type>Card Type</type>
    <number>New Account Number</number>
    <expDate>New Expiration Date</expDate>
  </newCardInfo>
</accountUpdater>
```

Example: accountUpdater Structure - Credit Cards with extendedCardResponse

```
<accountUpdater>
  <originalCardInfo>
    <type>Card Type</type>
    <number>Old Account Number</number>
    <expDate>Old Expiration Date</expDate>
  </originalCardInfo>
  <newCardInfo>
    <type>Card Type</type>
    <number>New Account Number</number>
    <expDate>New Expiration Date</expDate>
  </newCardInfo>
  <extendedCardResponse>
    <message>Code Description</message>
    <code>Either 501 or 504</code>
  </extendedCardResponse>
</accountUpdater>
```

Example: accountUpdater Structure - Credit Cards only extendedCardResponse

```
<accountUpdater>
  <extendedCardResponse>
    <message>Code Description</message>
    <code>Either 501 or 504</code>
  </extendedCardResponse>
</accountUpdater>
```

Example: accountUpdater Structure - Credit Cards (tokenized Merchant)

NOTE: This structure can also include the `<extendedCardResponse>` element.

```
<accountUpdater>
  <originalCardTokenInfo>
    <littleToken>Old Token</littleToken>
    <type>Card Type</type>
```



```
<expDate>Old Expiration Date</expDate>
<bin>Old Card BIN</bin>
</originalCardTokenInfo>
<newCardTokenInfo>
  <littleToken>New Token</littleToken>
  <type>Card Type</type>
  <expDate>New Expiration Date</expDate>
  <bin>New Card BIN</bin>
</newCardTokenInfo>
</accountUpdater>
```

Example: accountUpdater Structure - eCheck (for non-Tokenized Merchant)

```
<accountUpdater>
  <originalAccountInfo>
    <accType>Original Account Type</accType>
    <accNum>Original Account Number</accNum>
    <routingNum>Original Routing Number</routingNum>
  </originalAccountInfo>
  <newAccountInfo>
    <accType>New Account Type</accType>
    <accNum>New Account Number</accNum>
    <routingNum>New Routing Number</routingNum>
  </newAccountInfo>
</accountUpdateFileRequestData>
```

Example: accountUpdater Structure - eCheck (for Tokenized Merchant)

```
<accountUpdater>
  <originalTokenInfo>
    <accType>Original Account Type</accType>
    <littleToken>Original Token</littleToken>
    <routingNum>Original Routing Number</routingNum>
  </originalTokenInfo>
  <newTokenInfo>
    <accType>New Account Type</accType>
```

```
<littleToken>New Token</littleToken>  
<routingNum>New Routing Number</routingNum>  
</newTokenInfo>  
</accountUpdateFileRequestData>
```

4.7 accountUpdateResponse

The `accountUpdateResponse` element is the parent element for all Account Update responses transactions. You can use this only with Batch transactions.

Parent Elements:

[batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the accountUpdate transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the accountUpdate transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the accountUpdate transaction. minLength = 1 maxLength = 25

Child Elements: (Required)

[littleTxnId](#), [orderId](#), [response](#), [responseTime](#), [message](#)

Child Elements: (Optional)

[updatedCard](#), [originalCard](#), [originalToken](#), [updatedToken](#)

4.8 accType

The `accType` element is a required child of the `echeck`, `originalAccountInfo`, and `newAccountInfo` elements defining the type of eCheck account used in the transaction.

Type = Choice (Enum); **Enumerations** = Checking, Savings, Corporate, or Corp Savings

NOTE: Use Corporate for Corporate Checking accounts.

Parent Elements:

[echeck](#), [newAccountInfo](#), [originalAccountInfo](#), [originalTokenInfo](#), [newTokenInfo](#)

Attributes:

None

Child Elements:

None

4.9 actionReason

The `actionReason` element is an optional child of the `authReversal` element defining if the reversal is due to suspected fraud.

Type = String (Enum); **Enumerations** = SUSPECT_FRAUD

NOTE: When you include this optional element, the information will be forwarded to MasterCard as part of the MasterCard eCommerce Fraud Alert program.

Parent Elements:

[authReversal](#)

Attributes:

None

Child Elements:

None

4.10 addressIndicator

The `addressIndicator` element is an optional child of the `billMeLaterResponseData` element and indicates whether the shipping address is a commercial (**c**) or residential (**r**) shipping address.

Type = String; **minLength** = N/A; **maxLength** = 1

Parent Elements:

[billMeLaterResponseData](#)

Attributes:

None

Child Elements:

None

4.11 addressLine1, addressLine2, addressLine3

The elements `addressLine1`, `addressLine2`, and `addressLine3` define the address information in both the `billToAddress` and `shipToAddress` elements.

Type = String; **minLength** = N/A; **maxLength** = 35

Parent Elements:

[billToAddress](#), [shipToAddress](#)

Attributes:

None

Child Elements:

None

4.12 advancedAVSResults

The `advancedAVSResult` element is an optional child element of the `fraudResult` element. It defines the American Express Advanced AVS response codes that can be returned as verification of information supplied in the `<phone>` and/or `<email>` child elements of the `<billToAddress>` element. For a list of possible values, please refer to [AAVS Response Codes](#) on page 450.

NOTE: You must be certified to use LittleXML version 7.3 or above and specifically enabled to use the Advanced AVS feature. Please consult your Little Relationship Manager for additional information.

Type = String; minLength = N/A; maxLength = 3

Parent Elements:

[fraudResult](#)

Attributes:

None

Child Elements:

None

4.13 affiliate

The `affiliate` element is an optional child element of the `merchantData` element. You can use it to track transactions associated with various affiliate organizations.

NOTE: At present this data is not included in any reports. In the future it will be added to the Session export and the Net Settled Sales export reports.

Type = String; minLength = N/A; maxLength = 25

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [echeckCredit](#), [echeckSale](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements:

None

4.14 affluence

The `affluence` element is an optional child of the `enhancedAuthResponse` element and defines whether the card used falls into one of the two defined affluent categories. If the card does not meet the definition of either category, the system does not return the `affluence` element.

NOTE: Please consult your Relationship Manager for additional information concerning this feature.

Type = String (enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[enhancedAuthResponse](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
MASS AFFLUENT	Returned for certain Visa and MasterCard cards indicating high income customers (>100K annual income).
AFFLUENT	Returned for certain Visa and MasterCard cards indicating high income customers with high spending patterns (>100K annual income and >40K in card usage).

NOTE: The Affluence indicator applies only to certain Visa and MasterCard cards. This indicator does not apply to American Express or Discover cards.

4.15 allowPartialAuth

The `allowPartialAuth` element is an optional child of both `Authorization` and `Sale` transactions, which allows you to specify whether to authorize a partial amount if the entire requested authorization amount exceeds available credit/balance.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

None

NOTE: For a `Sale` transaction, the deposit will be for the partial amount.

4.16 amexAggregatorData

The `amexAggregatorData` element defines Amex Aggregator specific information in the Little XML. The system does not use the information unless you are designated as an Aggregator by American Express.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements: (Required)

[sellerId](#), [sellerMerchantCategoryCode](#)

Example: amexAggregatorData Structure

```
<amexAggregatorData>
  <sellerId>Seller Id</sellerId>
  <sellerMerchantCategoryCode>MerchantCategoryCode</sellerMerchantCategoryCode>
</amexAggregatorData>
```

4.17 amount

The `amount` element defines the amount of the transaction. Supply the value in cents without a decimal point. For example, a value of 1995 signifies \$19.95.

Type = Integer; **totalDigits** = 12

Parent Elements:

The `amount` element is a required child of each of the following Parent Elements: [authorization](#), [credit](#) (required if original transaction was not processed by Litle & Co.), [captureGivenAuth](#), [echeckCredit](#) (required if original transaction was not processed by Litle & Co.), [echeckSale](#), [echeckVerification](#), [forceCapture](#), [sale](#)

The `amount` element is an optional child of each of the following Parent Elements: [authReversal](#), [capture](#), [credit](#), [echeckCredit](#)

NOTE:	For all cases where the <code>amount</code> element is optional, if you do not specify a value, the system uses the entire amount from the referenced (by <code>littleTxnId</code>) transaction.
--------------	---

Attributes:

None

Child Elements:

None

4.18 approvedAmount

The `approvedAmount` element defines the dollar amount of the approval. It appears in an authorization or sale response only if the `allowPartialAuth` element is set to true in the request transaction.

Type = Integer; **totalDigits** = 8

Parent Elements:

[authorizationResponse](#), [saleResponse](#)

Attributes:

None

Child Elements:

None

4.19 authAmount

The `authAmount` element is an optional child of the `authInformation` element and is used to define the dollar amount of the authorization for Capture Given Auth transactions.

Type = Integer; **totalDigits** = 8

Parent Elements:

[authInformation](#)

Attributes:

None

Child Elements:

None

4.20 authCode

The `authCode` element is an optional child of both the `authorizationResponse` and `saleResponse` elements. It is also a required child of the `authInformation` element (used in `captureGivenAuth` transactions), where it specifies the authorization code from the associated Authorization or Sale transaction.

Type = String; **minLength** = N/A; **maxLength** = 6

Parent Elements:

[authInformation](#), [authorizationResponse](#), [saleResponse](#)

Attributes:

None

Child Elements:

None

4.21 authDate

The `authDate` element is a required child of the `authInformation` element and defines the date of the associated Authorization transaction.

Type = Date; **Format** = YYYY-MM-DD

Parent Elements:

[authInformation](#)

Attributes:

None

Child Elements:

None

4.22 authenticatedByMerchant

The `authenticatedByMerchant` element is an optional child element of the `cardholderAuthentication` element. This element indicates whether or not the customer has logged in to a secure web site or has been authenticated by the call center ANI.

For Bill Me Later transactions, set this element to **true** if this is an ecommerce transaction and you store and verify the customers BML account number. Set this element to **false** for call center BML transaction or ecommerce transactions if you do not store and verify the customers BML account number.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[cardholderAuthentication](#)

Attributes:

None

Child Elements:

None

4.23 authentication

The `authentication` element is a required element of both the `littleOnlineRequest` and the `batchRequest` elements. It contains child elements used to authenticate that the XML message is from a valid user.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

None

Child Elements:

Required: [user](#), [password](#)

Example: authentication Structure

```
<authentication>
  <user>User Name</user>
  <password>Password</password>
</authentication>
```

4.24 authenticationResult

The `authenticationResult` element is an optional child element of the `fraudResult` element. It defines the Visa CAVV Result code (from Verified by Visa). For a list of possible values, please refer to [Visa Authentication Result Codes](#) on page 447.

Type = String; **minLength** = N/A; **maxLength** = 1

Parent Elements:

[fraudResult](#)

Attributes:

None

Child Elements:

None

4.25 authenticationTransactionId

The `authenticationTransactionId` element is an optional child element of the `cardholderAuthentication` element. This element defines the Verified by Visa Transaction Id.

Type = Base 64 Encoded String; **minLength** = N/A; **maxLength** = 28

Parent Elements:

[cardholderAuthentication](#)

Attributes:

None

Child Elements:

None

4.26 authenticationValue

The `authenticationValue` element is an optional child element of the `cardholderAuthentication` element. This element defines either the Visa CAVV value (fixed length 28 characters) or the MasterCard UCAF value (variable length up to 32 characters).

Type = Base 64 Encoded String; **minLength** = N/A; **maxLength** = 32

Parent Elements:

[cardholderAuthentication](#)

Attributes:

None

Child Elements:

None

4.27 authInformation

The `authInformation` element is a required child of the `captureGivenAuth` element. It contains child elements used to provide details concerning the external (to the Little systems) Authorization.

Parent Elements:

[captureGivenAuth](#)

Attributes:

None

Child Elements:

Required: [authDate](#), [authCode](#)

Optional: [fraudResult](#), [authAmount](#)

Example: authInformation Structure

```
<authInformation>
  <authDate>Auth Date</authDate>
  <authCode>Approval Code</authCode>
  <fraudResult>
    <avsResult>AVS Verification Result Code</avsResult>
    <cardValidationResult>Validation Result Code</cardValidationResult>
    <authenticationResult>Verified by Visa Result Code</authenticationResult>
  </fraudResult>
  <authAmount>Amount of Authorization</authAmount>
</authInformation>
```

4.28 authorization

The `authorization` element is the parent element for all Authorization transactions. You can use this element in either Online or Batch transactions.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [amount](#), [orderSource](#), (choice of) [card](#), [paypal](#), [paypage](#), or [token](#), [cardholderAuthentication](#)

NOTE: The `cardholderAuthentication` child element is required only for 3-D Secure transactions.

Optional: [customerInfo](#), [billToAddress](#), [shipFromPostalCode](#), [billMeLaterRequest](#), [processingInstructions](#), [pos](#), [customBilling](#), [taxType](#), [enhancedData](#), [amexAggregatorData](#), [allowPartialAuth](#), [healthcareIIAS](#), [merchantData](#)

NOTE: The `enhancedData` element and two of its child elements, `deliveryType` and `shippingAmount`, are required for Bill Me Later Authorizations.

4.29 authorizationResponse

The `authorizationResponse` element is the parent element for information returned to you in response to an Authorization transaction. It can be a child of either a `litleOnlineResponse` element or a `batchResponse` element.

Parent Elements:

[litleOnlineResponse](#), [batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the Authorization transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the Authorization transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the Authorization transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [litleTxnId](#), [orderId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [cardProductId](#) (see Note below), [authCode](#), [authorizationResponseSubCode](#) (see Note below), [approvedAmount](#), [accountInformation](#), [fraudResult](#), [billMeLaterResponseData](#), [tokenResponse](#), [enhancedAuthResponse](#), [accountUpdater](#), [recycling](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

The `cardProductId` element returns a raw code referencing the card type. Please consult your Litle Relationship Manager for additional information.

The `authorizationResponseSubCode` element is not used at this time.

4.30 authorizationSourcePlatform

The `authorizationSourcePlatform` element is an optional child element of the `billMeLaterRequest` element. This element defines the physical platform that was used for submitting the authorization request (not the order). Specify as needed for auditing and re-authorization management purposes.

Type = String; **minLength** = N/A; **maxLength** = 1 (valid values below)

Value	Description
A	application processing
B	batch capture, recurring or mail order
C	call center
F	fulfillment/order management
K	kiosk
M	mobile device gateway
P	processor or gateway reauthorization
R	retail POS

Parent Elements:

[billMeLaterRequest](#)

Attributes:

None

Child Elements:

None

4.31 authReversal

The `authReversal` element is the parent element for all Authorization Reversal transactions. You can use this element in either Online or Batch transactions. Also see [Notes on the Use of Authorization Reversal Transactions](#) on page 36. Also, if you use the Little Recycling Engine, you can use the `authReversal` transaction to halt the recycling of an `authorization` transaction.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#)

Optional: [amount](#), [actionReason](#)

NOTE: If you do not specify an amount child element, the system reverses the full amount from the associated Authorization transaction.

4.32 authReversalResponse

The `authReversalResponse` element is the parent element for information returned to you in response to an Auth Reversal transaction. It can be a child of either a `littleOnlineResponse` element or a `batchResponse` element.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the authorization transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the authorization transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the authorization transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [orderId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.33 availableBalance

The `availableBalance` element is a required child element of the `fundingSource` element. It defines the outstanding available balance on the submitted prepaid card. If the balance can not be determined, the element returns, “Not Available.”

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[fundingSource](#)

Attributes:

None

Child Elements:

None

NOTE: The `fundingSource` element and its child elements, `type` and `availableBalance` are associated with the Smart Authorization features (see [Smart Authorization Features](#) on page 11.)

Please consult your Relationship Manager for additional information.

4.34 avsResult

The `avsResult` element is an optional child element of the `fraudResult` element. It defines the Address Verification response code returned by the networks. For a list of possible values, please refer to [AVS Response Codes](#) on page 449.

Type = String; **minLength** = N/A; **maxLength** = 2

Parent Elements:

[fraudResult](#)

Attributes:

None

Child Elements:

None

4.35 batchRequest

This is the root element for all LittleXML Batch requests.

Parent Elements:

[littleRequest](#)

Attributes:

NOTE: The xxxAmount attributes are required if the associated numXXX attribute is included and greater than 0. For example, if numAuths=5 and each Authorization is \$10.00, then you must include authAmount=5000.

Attribute Name	Type	Required?	Description
id	String	No	A unique string to identify this <code>batchRequest</code> within the Little system. minLength = N/A maxLength = 50
numAuths	Integer	No	Defines the total count of Authorization transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
authAmount	Integer	No	Defines the total dollar amount of Authorization transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numAuthReversals	Integer	No	Defines the total count of AuthReversal transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
authReversalAmount	Integer	No	Defines the total dollar amount of AuthReversal transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numCaptures	Integer	No	Defines the total count of Capture transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A

Attribute Name	Type	Required?	Description
captureAmount	Integer	No	Defines the total dollar amount of Capture transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numCredits	Integer	No	Defines the total count of Credit transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
creditAmount	Integer	No	Defines the total dollar amount of Credit transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numForceCaptures	Integer	No	Defines the total count of Force Capture transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
forceCaptureAmount	Integer	No	Defines the total dollar amount of Force Capture transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numSales	Integer	No	Defines the total count of Sale transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
saleAmount	Integer	No	Defines the total dollar amount of Sale transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numEmails (not used at this time)	Integer	No	Defines the total count of Email Notification transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
emailAmount (not used at this time)	Integer	No	Defines the total dollar amount of Email Notification transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10

Attribute Name	Type	Required?	Description
numCaptureGivenAuths	Integer	No	Defines the total count of Capture Given Auth transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
captureGivenAuthAmount	Integer	No	Defines the total dollar amount of Capture Given Auth transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numEcheckSales	Integer	No	Defines the total count of eCheck Sale transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
echeckSaleAmount	Integer	No	Defines the total dollar amount of eCheck Sale transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numEcheckCredit	Integer	No	Defines the total count of eCheck Credit transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
echeckCreditAmount	Integer	No	Defines the total dollar amount of eCheck Credit transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numEcheckVerification	Integer	No	Defines the total count of eCheck Verification transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
echeckVerificationAmount	Integer	No	Defines the total dollar amount of eCheck Verification transactions in the <code>batchRequest</code> . The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numEcheckRedeposit	Integer	No	Defines the total count of eCheck Redeposit transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
numAccountUpdates	Integer	No	Defines the total count of Account Update transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A

Attribute Name	Type	Required?	Description
numTokenRegistrations	Integer	No	Defines the total count of Token Registration transactions in the <code>batchRequest</code> . minLength = N/A maxLength = N/A
merchantId	String	Yes	A unique string to identify the merchant within the Little system. minLength = N/A maxLength = 50 Note: International currencies are supported on a per merchantId basis.

Child Elements:

Required: [authentication](#)

At least one of the following required: [authorization](#), [authReversal](#), [capture](#), [captureGivenAuth](#), [credit](#), [echeckCredit](#), [echeckRedeposit](#), [echeckSale](#), [echeckVerification](#), [forceCapture](#), [registerTokenRequest](#), [sale](#)

4.36 batchResponse

The `batchResponse` element is the parent element for information returned to you in response to a batch you submitted for processing. It is a child of a `littleResponse` element.

Parent Elements:

[littleResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the authorization transaction. minLength = N/A maxLength = 25
littleBatchId	Long	Yes	A unique value assigned by Little to identify the batch. minLength = N/A maxLength = 19
merchantId	String	Yes	The response returns the same value submitted in the authorization transaction. minLength = 1 maxLength = 50

Child Elements:

Required: [authorizationResponse](#), [authReversalResponse](#), [captureResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [echeckCreditResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [forceCaptureResponse](#), [registerTokenResponse](#), [saleResponse](#)

NOTE: The `batchResponse` contains child elements corresponding to the requests submitted in the `batchRequest`. For example, if the `batchRequest` contained 10 authorization and 8 capture transactions, the `batchResponse` would contain 10 `authorizationResponse` and 8 `captureResponse` transactions.

4.37 billMeLaterRequest

The `billMeLaterRequest` element is the parent of several child elements used to define merchant, product type, terms and conditions, and other items for Bill Me Later authorization transactions, or where you must reference an external (to Litle & Co.) BML transaction.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [sale](#)

Attributes:

None

Child Elements: (all optional)

[bmlMerchantId](#), [bmlProductType](#), [itemCategoryCode](#), [termsAndConditions](#), [preapprovalNumber](#), [virtualAuthenticationKeyPresenceIndicator](#), [virtualAuthenticationKeyData](#), [authorizationSourcePlatform](#)

NOTE: The following elements appear in the schema as children of the `billMeLaterRequest` element, but are not used at this time:

- `authorizationSourcePlatform`
 - `customerBillingAddressChanged`
 - `customerEmailChanged`
 - `customerPasswordChanged`
 - `customerPhoneChanged`
 - `merchantPromotionalCode`
 - `secretQuestionAnswer`
 - `secretQuestionCode`
-

Example: billMeLaterRequest

```
<billMeLaterRequest>
  <bmlMerchantId>bmlMerchantId</bmlMerchantId>
  <bmlProductType>bmlProductType</bmlProductType>
  <termsAndConditions>termsAndConditions</termsAndConditions>
  <preapprovalNumber>preapprovalNumber</preapprovalNumber>
  <virtualAuthenticationKeyPresenceIndicator> Indicator
</virtualAuthenticationKeyPresenceIndicator>
  <virtualAuthenticationKeyData> virtualAuthenticationKeyData
</virtualAuthenticationKeyData>
```

```
<itemCategoryCode>itemCategoryCode</itemCategoryCode>  
<authorizationSourcePlatform>platformType</authorizationSourcePlatform>  
</billMeLaterRequest>
```

4.38 billMeLaterResponseData

The `billMeLaterResponseData` element is the parent of several child elements used in the response XML for Bill Me Later authorization transactions.

Parent Elements:

[authorizationResponse](#), [saleResponse](#)

Attributes:

None

Child Elements:

[bmlMerchantId](#), [creditLine](#), [addressIndicator](#)

NOTE: The following elements appear in the schema as children of the `billMeLaterResponseData` element, but are not used at this time:

- `approvedTermsCode`
 - `loanToValueEstimator`
 - `promotionalCodeOffer`
 - `riskEstimator`
 - `riskQueueAssignment`
-

Example: billMeLaterResponseData Structure

```
<billMeLaterResponseData>
  <bmlMerchantId>bmlMerchantId</bmlMerchantId>
  <creditLine>creditLine</creditLine>
  <addressIndicator>addressIndicator</addressIndicator>
</billMeLaterResponseData>
```

4.39 billToAddress

The `billToAddress` element contains several child elements that define the postal mailing address (and telephone number) used for billing purposes. It also contains several elements used for the eCheck verification process.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [echeckCredit](#) (required if original transaction was not processed by Little & Co.), [echeckSale](#), [echeckVerification](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements: (all optional)

[name](#), [firstName](#), [middleInitial](#), [lastName](#), [companyName](#), [addressLine1](#), [addressLine2](#), [addressLine3](#), [city](#), [state](#), [zip](#), [country](#), [email](#), [phone](#)

NOTE: The `name` element is required for `echeckSale` and `echeckCredit` transactions. If you do not submit the customer name in one of these eCheck transactions, Little returns the response 330 - Invalid Payment Type.

For an `eCheckVerification` transaction, you must submit the `firstName`, `middleInitial` (optional), and `lastName` elements instead of the `name` element and the `companyName` element for a corporate account. You also must include the address, city, and state, and zip information. For a corporate account, if you do not have the name of the check issuer, you can use a value of "unavailable" for the `firstName` and `lastName` elements.

Example: billToAddress Structure

```
<billToAddress>
  <name>Customer's Full Name</name>
  <firstName>Customer's First Name</firstName>
  <middleInitial>Customer's First Name</middleInitial>
  <lastName>Customer's First Name</lastName>
  <companyName>Company's Name</companyName> (include for echeckVerification of
  corporate account)
  <addressLine1>Address Line 1</addressLine1>
  <addressLine2>Address Line 2</addressLine2>
  <addressLine3>Address Line 3</addressLine3>
  <city>City</city>
```

```
<state>State Abbreviation</state>  
<zip>Postal Code</zip>  
<country>Country Code</country>  
<email>Email Address</email>  
<phone>Telephone Number</phone>  
</billToAddress>
```


4.40 bin

The `bin` element provides the 6-digit Bank (or Issuer) Identification Number of the Issuing Bank. The system returns this value in XML responses when issuing new tokens to replace Visa or MasterCard account numbers. For Discover and American Express cards, this element is empty.

Type = String; **minLength** = 0; **maxLength** = 6

Parent Elements:

The `bin` element is an optional child of each listed parent element.

[registerTokenResponse](#), [tokenResponse](#), [newCardTokenInfo](#), [originalCardTokenInfo](#), [originalToken](#), [updatedToken](#)

Attributes:

None

Child Elements:

None

4.41 bmlMerchantId

The `bmlMerchantId` element is a value assigned by Bill Me Later to identify the merchant within the Bill Me Later system.

Type = Long; **minLength** = N/A; **maxLength** = 19

Parent Elements:

The `bmlMerchantId` element is an optional child of each listed parent element.

[billMeLaterRequest](#), [billMeLaterResponseData](#)

Attributes:

None

Child Elements:

None

4.42 bmlProductType

The `bmlProductType` element is a value assigned by Bill Me Later to identify the merchant account type within the Bill Me Later system.

Type = Long; **minLength** = N/A; **maxLength** = 19

Parent Elements:

The `bmlProductType` element is an optional child of each listed parent element.

[billMeLaterRequest](#), [billMeLaterResponseData](#)

NOTE: At this time, the only valid value for this element is BL.

Attributes:

None

Child Elements:

None

4.43 bypassVelocityCheck

The `bypassVelocityCheck` element is an optional child of the `processingInstructions` element, which allows you to specify whether or not the system performs velocity checking on the transaction.

NOTE: Velocity Checking is not currently supported.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[processingInstructions](#)

Attributes:

None

Child Elements:

None

4.44 campaign

The `campaign` element is an optional child element of the `merchantData` element. You can use it to track transactions associated with various marketing campaigns.

NOTE: At present this data is not included in any reports. In the future it will be added to the Session export and the Net Settled Sales export reports.

Type = String; minLength = N/A; maxLength = 25

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [echeckCredit](#), [echeckSale](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements:

None

4.45 capability

The `capability` element is a required child of the `pos` element, which describes the capability of the point of sale terminal.

Type = String (Enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[pos](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
notused	terminal not used
magstripe	magnetic stripe reader capability
keyedonly	keyed entry only capability

4.46 capture

The capture element is the parent element for all Capture (deposit) transactions. You can use this element in either Online or Batch transactions.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. This attribute is also used for Duplicate Transaction Detection. Failure to include a value for this attribute disables Duplicate Detection for the transaction. Please refer to Duplicate Transaction Detection on page 6 for additional information about the operation of Duplicate checking. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25
partial	Boolean	No	If there is more than one capture that references the same <code><littleTxnId></code> , set this attribute to “true” for each of those partial captures. The default value is false. Valid Values = true or false

Child Elements:

Required: [littleTxnId](#), [paypalOrderComplete](#) (required only if closing a PayPal order)

Optional: [amount](#), [enhancedData](#), [processingInstructions](#),

NOTE: If you do not specify an amount child element, the system uses the full amount from the associated Authorization transaction.

4.47 captureGivenAuth

The `captureGivenAuth` element is the parent element for all Capture Given Auth transactions. These are specialized Capture transactions used when the `littleTxnId` for the associated Authorization is unknown or when the Authorization occurred outside the Little & Co. system. You can use this element in either Online or Batch transactions.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. This attribute is also used for Duplicate Transaction Detection. Failure to include a value for this attribute disables Duplicate Detection for the transaction. Please refer to Duplicate Transaction Detection on page 6 for additional information about the operation of Duplicate checking. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [authInformation](#), [amount](#), [orderSource](#), choice of [card](#), [token](#), or [paypage](#)

Optional: [billToAddress](#), [shipFromPostalCode](#), [customBilling](#), [taxType](#), [enhancedData](#), [processingInstructions](#), [pos](#), [amexAggregatorData](#), [merchantData](#)

NOTE: If you do not specify an amount child element, the system uses the full amount from the associated Authorization transaction.

4.48 captureGivenAuthResponse

The `captureGivenAuthResponse` element is the parent element for information returned to you in response to a Capture Given Auth transaction. It can be a child of either a `littleOnlineResponse` element or a `batchResponse` element.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the Capture Given Auth transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the Capture Given Auth transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the Capture Given Auth transaction. minLength = 1 maxLength = 25
duplicate	Boolean	No	If the request is a duplicate (see Online Duplicate Checking on page 7), the response includes the duplicate flag set to true and the entire original response. Note: This attribute applies only to Online transaction responses.

Child Elements:

Required: [littleTxnId](#), [orderId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [tokenResponse](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.49 captureResponse

The `captureResponse` element is the parent element for information returned to you in response to a Capture transaction. It can be a child of either a `littleOnlineResponse` element or a `batchResponse` element.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the capture transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the capture transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the capture transaction. minLength = 1 maxLength = 25
duplicate	Boolean	No	If the request is a duplicate (see Online Duplicate Checking on page 7), the response includes the duplicate flag set to true and the entire original response. Note: This attribute applies only to Online transaction responses.

Child Elements:

Required: [littleTxnId](#), [orderId](#) (required for Batch), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [accountUpdater](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.50 card

The `card` element defines payment card information. It is a required element for most transaction types unless the transaction uses an alternate payment method such as PayPal. It contains one or more child elements depending upon whether the transaction is a card-not-present or a card-present (face-to-face) transaction.

Parent Elements:

[accountUpdate](#), [authorization](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements:

For card-not-present transactions (Required): [type](#), [number](#), [expDate](#)

For card-present transactions (Required): [track](#)

For both transactions types (Optional): [cardValidationNum](#)

Example: card Structure - Card-Not-Present

```
<card>
  <type>Card Type Abbreviation</type>
  <number>Account Number</number>
  <expDate>Expiration Date</expDate>
  <cardValidationNum>Card Validation Number</cardValidationNum>
</card>
```

Example: card Structure - Card-Present

```
<card>
  <track>Magnetic Stripe Read</track>
</card>
```

4.51 cardAcceptorTaxId

The `cardAcceptorTaxId` element is an optional child of the `detailTax` element and defines the merchant's Tax Id. This ID is nine digits long if the merchant is domiciled in the U.S. If the card acceptor tax ID is unknown, do not include this element.

Type = String; **minLength** = 1; **maxLength** = 20

Parent Elements:

[detailTax](#)

Attributes:

None

Child Elements:

None

4.52 cardholderAuthentication

The `cardholderAuthentication` element is an optional child element of the Authorization and Sale transactions. The children of this element have two purposes. The first is to define Verified by Visa or MasterCard SecureCode data in the Authorization or Sale transactions (`authenticationValue` and `authenticationTransactionId` elements). The remaining child elements, `customerIpAddress` and `authenticatedByMerchant`, are used in Bill Me Later transactions. The `customerIpAddress` element can also be used to supply the customer IP Address by merchants enabled for American Express Advanced AVS services.

NOTE: The `customerIpAddress` child element is required for BML ecommerce transactions to succeed.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

[authenticationValue](#), [authenticationTransactionId](#), [customerIpAddress](#), [authenticatedByMerchant](#)

Example: cardholderAuthentication Structure

```
<cardholderAuthentication>
  <authenticationValue>BwABBJQ1AgAAAAAgJDUCAAAAAA=</authenticationValue>
  <authenticationTransactionId>EaOMucALHQqLAEGAgk=</authenticationTransactionId>
  <customerIpAddress>Customer Ip</customerIpAddress>
  <authenticatedByMerchant>Boolean</authenticatedByMerchant>
</cardholderAuthentication>
```

NOTE: The values for the `<authenticationValue>` and `<authenticationTransactionId>` elements in the example above have been truncated.

4.53 cardholderId

The `cardholderId` element is a required child of the `pos` element, which describes the method used for cardholder identification at the point of sale.

Type = String (Enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[pos](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
signature	customer signature obtained
pin	PIN number
nopin	unattended terminal - no PIN pad
directmarket	mail, telephone, or online

4.54 cardOrToken

The `cardOrToken` element is an abstract that allows the substitution of either the `card` or `token` element. You must specify one of the two substitution elements as a child of the `accountUpdate` element.

Parent Elements:

[accountUpdate](#)

Substitution Options:

[card](#), [token](#)

4.55 cardProductType

The `cardProductType` element is an optional child of the `enhancedAuthResponse` element and whether the card used is commercial or consumer.

Type = String (enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[enhancedAuthResponse](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
COMMERCIAL	The card is a commercial card.
CONSUMER	The card is a consumer card.
UNKNOWN	The type of card is not known.

4.56 cardValidationNum

The `cardValidationNum` element is an optional child of the `card` element, which you use to submit either the CVV2 (Visa), CVC2 (MasterCard), or CID (American Express and Discover) value.

NOTE: Some American Express cards may have a 4-digit CID on the front of the card and/or a 3-digit CID on the back of the card. You can use either of the numbers for card validation, but not both.

Type = String; minLength = N/A; maxLength = 4

Parent Elements:

[card](#), [paypage](#), [token](#)

Attributes:

None

Child Elements:

None

4.57 cardValidationResult

The `cardValidationResult` element is an optional child element of the `fraudResult` element. It defines the Card Validation response code returned by the networks. For a list of possible values, please refer to [Card Validation Response Codes](#) on page 453.

Type = String; **minLength** = N/A; **maxLength** = 2

Parent Elements:

[fraudResult](#)

Attributes:

None

Child Elements:

None

4.58 chargeback

The `chargeback` element is an optional child of the `filtering` element. To disable the filtering operation for a selected transaction include the `chargeback` element with a setting of **false**.

Type = Boolean; **Valid Value** = false

Parent Elements:

[filtering](#)

Attributes:

None

Child Elements:

None

4.59 checkNum

The `checkNum` element is an optional child of the `echeck` element defining the check number of used in the transaction.

Type = String; **minLength** = N/A; **maxLength** = 15

Parent Elements:

[echeck](#)

Attributes:

None

Child Elements:

None

4.60 city

The `city` element defines the customer's city name in the `billToAddress` and `shipToAddress` elements. In the `customBilling` element, `city` defines the location of the merchant for card-present transactions.

Type = String; **minLength** = N/A; **maxLength** = 35

Parent Elements:

[billToAddress](#), [shipFromPostalCode](#), [customBilling](#)

Attributes:

None

Child Elements:

None

4.61 clinicOtherAmount

The `clinicAmount` element is an optional child of the `healthcareAmounts` element and defines the healthcare amount used for the clinic/office visits. The decimal is implied. Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

Optional: [healthcareAmounts](#)

Attributes:

None

Child Elements:

None

4.62 code

The `code` element is a required child of the `extendedCardResponse` element. The `code/message` combination can be either 501- The account was closed, or 504 - Contact the cardholder for updated information.

Type = String; **minLength** = N/A; **maxLength** = 3

Parent Elements:

[extendedCardResponse](#)

Attributes:

None

Child Elements:

None

4.63 commodityCode

The `commodityCode` element is an optional child of the `lineItemData` element, which specifies the Identifier assigned by the card acceptor that categorizes the purchased item. Although the schema defines it as an optional child of the `enhancedData` element, it is required by Visa for Level III interchange rates.

Type = String; **minLength** = 1; **maxLength** = 12

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

NOTE: A commodity code is a numeric code representing a particular product or service. The code can be 3, 5, 7, or 11 digits in length. The longer the code the more granular the description of the product/service. For example, code 045 is used for Appliances and Equipment, Household Type, while code 04506 represents the sub-set of Appliances, Small Electric.

The codes are issued by the NIGP (National Institute of Governmental Purchasing). Their site, www.ngip.com, offers a subscription based code search engine, as well as downloadable lists for purchase.

You can also find many lists published online by performing a simple search on “Commodity Codes”.

4.64 companyName

The `companyName` element is an optional child of the `billToAddress` element, which specifies the name of the company associated with the corporate checking account. This element is required when performing an eCheck Verification of a check from a corporate account, as defined by the `<accType>` child of the `<echeck>` element.

Type = String; **minLength** = N/A; **maxLength** = 40

Parent Elements:

[billToAddress](#)

Attributes:

None

Child Elements:

None

4.65 country

The `country` element defines the country portion of the postal mailing address in both the `billToAddress` and `shipToAddress` elements.

Type = String (Enum); **minLength** = N/A; **maxLength** = 3

NOTE: The enumerations for this element are listed under `<countryTypeEnum>` in the LittleXML Common XSD. The country names corresponding to the abbreviations can be found in the ISO 3166-1 standard.

Parent Elements:

[billToAddress](#), [shipFromPostalCode](#)

Attributes:

None

Child Elements:

None

4.66 credit

The `credit` element is the parent element for all Credit transactions. You can use this element in either Online or Batch transactions.

Parent Elements:

[litleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. This attribute is also used for Duplicate Transaction Detection. Failure to include a value for this attribute disables Duplicate Detection for the transaction. Please refer to Duplicate Transaction Detection on page 6 for additional information about the operation of Duplicate checking. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

For credits to transactions processed by Litle & Co. (Required): [litleTxnId](#)

For credits to transactions processed by Litle & Co. (Optional): [amount](#)

NOTE: If you do not specify an `amount` child element, the system uses the full amount from the associated Capture, Force Capture, or Sale transaction. The `amount` element is required for credits to transactions not processed by Litle & Co.

For credits to transactions not processed by Litle & Co. (Required): [orderId](#), [amount](#), [orderSource](#), choice of [card](#), [token](#), [paypage](#), or paypal

For credits to transactions not processed by Litle & Co. (Optional): [billToAddress](#), [billMeLaterRequest](#), [pos](#), [amexAggregatorData](#)

For both transaction types (Optional): [customBilling](#), [taxType](#), [enhancedData](#), [processingInstructions](#), [merchantData](#)

4.67 creditLine

The `creditLine` element is an optional child of the `billMeLaterResponseData` element and indicates the credit line of the customer. The amount is specified using a two-digit implied decimal.

Type = Integer; **totalDigits** = 8

Parent Elements:

[billMeLaterResponseData](#)

Attributes:

None

Child Elements:

None

4.68 creditResponse

The `creditResponse` element is the parent element for information returned to you in response to a Credit transaction. It can be a child of either a `littleOnlineResponse` element or a `batchResponse` element.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the credit transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the credit transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the credit transaction. minLength = 1 maxLength = 25
duplicate	Boolean	No	If the request is a duplicate (see Online Duplicate Checking on page 7), the response includes the duplicate flag set to true and the entire original response. Note: This attribute applies only to Online transaction responses.

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [orderId](#), [tokenResponse](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.69 customBilling

The customBilling element allows you to specify custom billing descriptor information for the transaction. This billing descriptor is used instead of the descriptor defined as the default billing descriptor. If you do not define this element, the default is used.

NOTE: If you submit a captureGivenAuth transaction with a customBilling element and a matching Authorization is found (see [Capture Given Auth Transaction](#) on page 38), the system uses the customBilling information from the Authorization and discards the information from the captureGivenAuth.

Parent Elements: (all Optional)

[authorization](#), [captureGivenAuth](#), [credit](#), [echeckCredit](#), [echeckSale](#), [forceCapture](#), [sale](#),

Attributes:

None

Child Elements:

Required for card-not-present transactions: [phone](#), or [url](#)

NOTE: Please consult your Little Relationship Manager prior to using the <url> element. The contents of this element are discarded unless you are specifically enabled to use it in your LittleXML submissions.

Required for card present transactions: [city](#)

Optional for either: [descriptor](#)

Example: customBilling Structure - Card-Not-Present (using phone child)

```
<customBilling>
  <phone>Telephone Number</phone>
  <descriptor>Billing Descriptor</descriptor>
</customBilling>
```

Example: customBilling Structure - Card-Not-Present (using url child)

```
<customBilling>
  <url>retail.url</url>
  <descriptor>www.retail.com</descriptor>
</customBilling>
```

Example: customBilling Structure - Card-Present

```
<customBilling>  
  <city>City</city>  
  <descriptor>Billing Descriptor</descriptor>  
</customBilling>
```


4.70 customerInfo

The `customerInfo` element is the parent of several child elements use to define customer information for Bill Me Later transactions.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

[ssn](#), [dob](#), [customerRegistrationDate](#), [customerType](#), [incomeAmount](#), [employerName](#), [customerWorkTelephone](#), [residenceStatus](#), [yearsAtResidence](#), [yearsAtEmployer](#)

NOTE: Although the schema defines all child elements as optional, under certain conditions `ssn`, `dob`, `customerRegistrationDate`, and `customerType` are required for the transaction to succeed.

Example: customerInfo Structure

```
<customerInfo>
  <ssn>ssn</ssn>
  <dob>dob</dob>
  <customerRegistrationDate>customerRegistrationDate</customerRegistrationDate>
  <customerType>customerType</customerType>
  <incomeAmount>incomeAmount</incomeAmount>
  <incomeCurrency>incomeCurrency</incomeCurrency>
  <employerName>employerName</employerName>
  <customerWorkTelephone>customerWorkTelephone</customerWorkTelephone>
  <residenceStatus>residenceStatus</residenceStatus>
  <yearsAtResidence>yearsAtResidence</yearsAtResidence>
  <yearsAtEmployer>yearsAtEmployer</yearsAtEmployer>
</customerInfo>
```

4.71 customerIpAddress

The `customerIpAddress` element is an optional child element of the `cardholderAuthentication` element. This element defines the IP Address of the customer's system. This element is used either for Bill Me Later transactions or to supply the customer IP Address by merchants enables for American Express Advanced AVS services.

Type = Ip Address; **Format** = nnn.nnn.nnn.nnn

NOTE: This element is required for BML ecommerce transactions to succeed.

Parent Elements:

[cardholderAuthentication](#)

Attributes:

None

Child Elements:

None

4.72 customerReference

The `customerReference` element defines a reference string used by the customer for the purchase (for example, a Purchase Order Number). Although the schema defines it as an optional child of the `enhancedData` element, it is required by Visa for Level III interchange rates; however, you should omit this element if it is blank.

Type = String; **minLength** = 1; **maxLength** = 17

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.73 customerRegistrationDate

The `customerRegistrationDate` element is an optional child of the `customerInfo` element defining the earliest date on file with this customer. The latest allowable date is the current date. It is used in combination with several other elements to provide required information for some Bill Me Later transactions.

Type = Date; **Format** = YYYY-MM-DD

NOTE: In order for a BML transaction to succeed, you must include this element if the customer does not have a BML account.

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.74 customerType

The `customerType` element is an optional child of the `customerInfo` element defining whether the customer is a new or existing customer. An existing customer is a customer in good standing that has been registered with the merchant for a minimum of 30 days and has made at least one purchase in the last 30 days. It is used in combination with several other elements to provide required information for some Bill Me Later transactions.

Type = Choice (Enum); **Enumerations** = New or Existing

NOTE: In order for a BML transaction to succeed, you must include this element if the customer does not have a BML account.

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.75 customerWorkTelephone

The `customerWorkTelephone` element is an optional child of the `customerInfo` element and defines the customer's work telephone number. It is used in combination with several other elements to provide information for some Bill Me Later transactions.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.76 deliveryType

The `deliveryType` element is an optional child of the `enhancedData` element and defines the shipping method used for delivery of the product.

NOTE: Although define in the schema as an optional child of the `enhancedData` element, `deliveryType` is required for Bill Me Later transactions.

Type = String (enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
CNC	Cash and Carry
DIG	Digital Delivery
PHY	Physical Delivery
SVC	Service Delivery
TBD (default)	To be determined.

4.77 dentalAmount

The `dentalAmount` element is an optional child of the `healthcareAmounts` element and defines the healthcare amount used for dental related purchases. The decimal is implied.

Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

Optional: [healthcareAmounts](#)

Attributes:

None

Child Elements:

None

4.78 descriptor

The `descriptor` element is a required child of the `customBilling` element. This element defines the text you wish to display on the customer bill, enabling the customer to better recognize the charge.

Type = String; **minLength** = N/A; **maxLength** = 25

NOTE: If you include a prefix:

- the prefix must be either 3, 7, or 12 characters in length.
 - you must use an asterisk (*) after the prefix as a separator, in one of the following positions: 4th, 8th, or 13th. Do not use an asterisk in more than one position.
 - Use only the following valid characters:
 - Numbers
 - Letters
 - Special characters as follows: ampersand, asterisk (Required; see note above), comma, dash, period, or pound sign.
-

Parent Elements:

[customBilling](#)

Attributes:

None

Child Elements:

None

4.79 destinationCountryCode

The `destinationCountryCode` element defines the country portion of the postal mailing address in the `enhancedData` element.

Type = String (Enum); **minLength** = N/A; **maxLength** = 3

NOTE:	The enumerations for this element are listed under <code><countryTypeEnum></code> in the LittleXML Common XSD. The country names corresponding to the abbreviations can be found in the ISO 3166-1 standard.
--------------	--

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.80 destinationPostalCode

The `destinationPostalCode` element defines the postal code of the destination in the `enhancedData` element.

Type = String; **minLength** = N/A; **maxLength** = 20

NOTE: Although the schema specifies the **maxLength** of the `<destinationPostalCode>` element as 20 characters, in practice you should never exceed 10 characters in your submissions.

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.81 detailTax

The `detailTax` element is an optional child of both the `enhancedData` and `lineItemData` elements, which you use to specify detailed tax information (for example, city or local tax). The total sum of the `detailTax` values should match either the `salesTax` value, if `detailTax` is a child of `enhancedData`, or the `taxAmount` element if `detailTax` is a child of `lineItemData`.

Parent Elements:

[enhancedData](#), [lineItemData](#)

NOTE: The `detailTax` element can appear a maximum of six times as a child of either parent.

Attributes:

None

Child Elements:

Required: [taxAmount](#)

Optional: [taxIncludedInTotal](#), [taxRate](#), [taxTypeIdentifier](#), [cardAcceptorTaxId](#)

Example: detailTax Structure

```
<detailTax>
  <taxIncludedInTotal>true or false</taxIncludedInTotal>
  <taxAmount>Additional Tax Amount</taxAmount>
  <taxRate>Tax Rate of This Tax Amount</taxRate>
  <taxTypeIdentifier>Tax Type Enum</taxTypeIdentifier>
  <cardAcceptorTaxId>Tax ID of Card Acceptor</cardAcceptorTaxId>
</detailTax>
```

4.82 discountAmount

The `discountAmount` element defines the amount of the discount for the order. Although the schema defines it as an optional child of the `enhancedData` element, it is required by Visa for Level III interchange rates. The decimal is implied. Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.83 dob

The `dob` element is an optional child of the `customerInfo` element. It is used in combination with several other elements to provide required information for some Bill Me Later transactions.

Type = Date; **Format** = YYYY-MM-DD

NOTE:	In order for a BML transaction to succeed, you must include this element if: <ul style="list-style-type: none">• the customer does not have a BML account or <ul style="list-style-type: none">• the customer has a BML account, but the account has not been authenticated. You do not need to include this element if the BML account has been authenticated.
--------------	--

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.84 dutyAmount

The `dutyAmount` element defines duty on the total purchased amount for the order. Although the schema defines it as an optional child of the `enhancedData` element, it is required by Visa for Level III interchange rates. The decimal is implied. Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.85 echeck

The `echeck` element is a required child of the `echeckSale`, `echeckVerification`, and `echeckCredit` (when the credit is against a transaction not originally processed through the Little & Co. system) elements. It contains child elements used to provide details concerning the eCheck account.

Parent Elements:

[echeckCredit](#), [echeckSale](#), [echeckVerification](#)

Attributes:

None

Child Elements:

Required: [accType](#), [accNum](#), [routingNum](#)

Optional: [checkNum](#)

Example: echeck Structure

```
<echeck>
  <accType>Account Type Abbreviation</accType>
  <accNum>Account Number</accNum>
  <routingNum>Routing Number</routingNum>
  <checkNum>Check Number</checkNum>
</echeck>
```


4.86 eCheckAccountSuffix

The `eCheckAccountSuffix` element is an optional child of the `tokenResponse` element that provides the last three characters of the eCheck account number.

Type = String; **minLength** = 3; **maxLength** = 3

Parent Elements:

[registerTokenResponse](#), [tokenResponse](#)

Attributes:

None

Child Elements:

None

4.87 echeckCredit

The `echeckCredit` element is the parent element for all eCheck Credit transactions. You can use this element in either Batch or Online transactions.

Parent Elements:

[batchRequest](#), [litleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. This attribute is also used for Duplicate Transaction Detection. Failure to include a value for this attribute disables Duplicate Detection for the transaction. Please refer to Duplicate Transaction Detection on page 6 for additional information about the operation of Duplicate checking. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

For credits to transactions processed by Litle & Co. (Required): [litleTxnId](#)

For credits to transactions processed by Litle & Co. (Optional): [amount](#)

NOTE: If you do not specify an amount child element, the system uses the full amount from the associated `echeckSale` transaction.

For credits to transactions not processed by Litle & Co. (Required): [orderId](#), [amount](#), [orderSource](#), [billToAddress](#), [echeckOrEcheckToken](#) (allows the substitution of either the `echeck` or `echeckToken` elements)

For credits to transactions not processed by Litle & Co. (Optional): [merchantData](#)

For both (Optional): [customBilling](#)

4.88 echeckCreditResponse

The `echeckCreditResponse` element is the parent element for information returned to you in response to an `echeckCredit` transaction.

Parent Elements:

[batchResponse](#), [littleOnlineResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the <code>echeckCredit</code> transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the <code>echeckCredit</code> transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the <code>echeckCredit</code> transaction. minLength = 1 maxLength = 25
duplicate	Boolean	No	If the request is a duplicate (see Online Duplicate Checking on page 7), the response includes the duplicate flag set to true and the entire original response. Note: This attribute applies only to Online transaction responses.

Child Elements: (all Required)

[littleTxnId](#), [orderId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [accountUpdater](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.89 echeckForToken

The `echeckForToken` element is a child of the `registerTokenRequest` element. It contains the routing and account number of the eCheck account which the system uses to generate a token.

Parent Elements:

[registerTokenRequest](#)

Attributes:

None

Child Elements:

Required: [accNum](#), [routingNum](#)

Example: echeck Structure

```
<echeckForToken>
  <accNum>Account Number</accNum>
  <routingNum>Routing Number</routingNum>
</echeckForToken>
```

4.90 echeckOrEcheckToken

The `echeckOrEcheckToken` element is an abstract that allows the substitution of either the `echeck` or `echeckToken` element. In eCheck transactions, except `echeckVoid`, you must specify one of the two substitution elements as a child.

Parent Elements:

[echeckCredit](#), [echeckRedeposit](#), [echeckSale](#), [echeckVerification](#)

Substitution Options:

[echeck](#), [echeckToken](#)

4.91 echeckRedeposit

The `echeckRedeposit` element is the parent element for all eCheck Redeposit transactions. You use this transaction type to manually attempt redeposits of eChecks returned for either Insufficient Funds or Uncollected Funds. You can use this element in either Batch or Online transactions.

NOTE: Do not use this transaction type if you are enabled for the Auto Redeposit feature. If you are enabled for the Auto Redeposit feature, the system will reject any `echeckRedeposit` transaction you submit.

Parent Elements:

[batchRequest](#), [litleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [litleTxnId](#)

Optional: [echeckOrEcheckToken](#) (allows the substitution of either the `echeck` or `echeckToken` elements)

4.92 echeckRedepositResponse

The `echeckRedepositResponse` element is the parent element for information returned to you in response to an `echeckRedeposit` transaction.

Parent Elements:

[batchResponse](#), [litleOnlineResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the <code>echeckSale</code> transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the <code>echeckSale</code> transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the <code>echeckSale</code> transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [litleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [accountUpdater](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.93 echeckSale

The echeckSale element is the parent element for all eCheck Sale transactions. You can use this element in either Batch or Online transactions.

Parent Elements:

[batchRequest](#), [litleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. This attribute is also used for Duplicate Transaction Detection. Failure to include a value for this attribute disables Duplicate Detection for the transaction. Please refer to Duplicate Transaction Detection on page 6 for additional information about the operation of Duplicate checking. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [amount](#), [orderSource](#), [billToAddress](#), [echeckOrEcheckToken](#) (allows the substitution of either the `echeck` or `echeckToken` elements)

Optional: [shipToAddress](#), [verify](#), [customBilling](#), [merchantData](#)

4.94 echeckSalesResponse

The `echeckSalesResponse` element is the parent element for information returned to you in response to an `echeckSale` transaction.

Parent Elements:

[batchResponse](#), [litleOnlineResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the <code>echeckSale</code> transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the <code>echeckSale</code> transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the <code>echeckSale</code> transaction. minLength = 1 maxLength = 25
duplicate	Boolean	No	If the request is a duplicate (see Online Duplicate Checking on page 7), the response includes the duplicate flag set to true and the entire original response. Note: This attribute applies only to Online transaction responses.

Child Elements:

Required: [litleTxnId](#), [orderId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [accountUpdater](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.95 echeckToken

The `echeckToken` element replaces the `echeck` element in tokenized eCheck transactions and defines the tokenized account information.

Parent Elements:

[echeckCredit](#), [echeckRedeposit](#), [echeckSale](#), [echeckVerification](#)

Attributes:

None

Child Elements:

Required: [litleToken](#), [routingNum](#), [accType](#)

Optional: [checkNum](#)

Example: echeck Structure

```
<echeckToken>
  <litleToken>Little Token</litleToken>
  <routingNum>Routing Number</routingNum>
  <accType>Account Type Abbreviation</accType>
  <checkNum>Check Number</checkNum>
</echeckToken>
```

4.96 echeckVerification

The `echeckVerification` element is the parent element for all eCheck Verification transactions. You use this transaction type to initiate a comparison of the consumer's account information against positive/negative databases. You can use this element in either Batch or Online transactions.

Parent Elements:

[batchRequest](#), [littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [amount](#), [orderSource](#), [billToAddress](#), [echeckOrEcheckToken](#) (allows the substitution of either the `echeck` or `echeckToken` elements)

Optional: [littleTxnId](#)

4.97 echeckVerificationResponse

The `echeckVerificationResponse` element is the parent element for information returned to you in response to a eCheck Verification transaction.

Parent Elements:

[batchResponse](#), [littleOnlineResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the capture transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the capture transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the capture transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [orderId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.98 echeckVoid

The `echeckVoid` element is the parent element for all eCheck Void transactions. You use this transaction type to either cancel an eCheck Sale transaction, as long as the transaction has not yet settled, or halt automatic redeposit attempts of eChecks returned for either Insufficient Funds or Uncollected Funds. You can use this element only in Online transactions.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. This attribute is also used for Duplicate Transaction Detection. Failure to include a value for this attribute disables Duplicate Detection for the transaction. Please refer to Duplicate Transaction Detection on page 6 for additional information about the operation of Duplicate checking. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#)

4.99 echeckVoidResponse

The `echeckVoidResponse` element is the parent element for information returned to you in response to an eCheck Void transaction.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the void transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the void transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the void transaction. minLength = 1 maxLength = 25
duplicate	Boolean	No	If the request is a duplicate (see Online Duplicate Checking on page 7), the response includes the duplicate flag set to true and the entire original response. Note: This attribute applies only to Online transaction responses.

Child Elements: (all Required)

[littleTxnId](#), [response](#), [responseTime](#), [message](#), [postDate](#)

4.100 email

The `email` element defines the email address of the customer in both the `billToAddress` and `shipToAddress` elements.

Type = String; **minLength** = N/A; **maxLength** = 100

Parent Elements:

[billToAddress](#), [shipToAddress](#)

Attributes:

None

Child Elements:

None

4.101 employerName

The `employerName` element is an optional child of the `customerInfo` element and defines the name of the customer's place of employment. It is used in combination with several other elements to provide information for some Bill Me Later transactions.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.102 enhancedAuthResponse

The `enhancedAuthResponse` element is an optional child of both the `authorizationResponse` and `saleResponse` elements. Through its child elements, it can provide information concerning whether the card used for the transaction is Prepaid and if so, the available balance. If the card used meets certain other criteria, an affluence indicator can be included

NOTE: The `enhancedAuthResponse` includes the `fundingSource` child only if the card is identified as Prepaid.
The `enhancedAuthResponse` includes the `affluence` child only if the card is identified as meeting the criteria for either a `Affluent` or `Mass Affluent` card.

Parent Elements:

[authorizationResponse](#), [saleResponse](#)

Attributes:

None

Child Elements: (all Optional)

[fundingSource](#), [affluence](#), [issuerCountry](#), [cardProductType](#)

Example: enhancedAuthResponse - with fundingSource

```
<enhancedAuthResponse>
  <fundingSource>
    <type>PREPAID</type>
    <availableBalance>0</availableBalance>
    <reloadable>YES|NO|UNKNOWN</reloadable>
    <prepaidCardType>GIFT</prepaidCardType>
  </fundingSource>
</enhancedAuthResponse>
```

Example: enhancedAuthResponse - with affluence

```
<enhancedAuthResponse>
  <affluence>AFFLUENT</affluence>
</enhancedAuthResponse>
```

Example: enhancedAuthResponse - with issuerCountry

```
<enhancedAuthResponse>  
  <issuerCountry>MEX</issuerCountry>  
</enhancedAuthResponse>
```

Example: enhancedAuthResponse - with cardProductType

```
<enhancedAuthResponse>  
  <cardProductType>CONSUMER</cardProductType>  
</enhancedAuthResponse>
```

4.103 enhancedData

The `enhancedData` element allows you to specify extra information concerning a transaction in order to qualify for certain purchasing interchange rates as follows:

TABLE 4-1 Level 2 and Level 3 Data Requirements

Method of Payment	Level 2 Requirements	Level 3 Requirements
Visa	<ol style="list-style-type: none"> 1. The transaction must be taxable. 2. The tax charged must be between 0.1% and 22% of the transaction amount. 	<ol style="list-style-type: none"> 1. The transaction must use a purchasing card. 2. You must include at least one line item with amount, description, and quantity defined.
MasterCard	<ol style="list-style-type: none"> 1. The transaction must be taxable 2. Tax charged must be between 0.1% and 30% of the transaction amount. <p>Note: You can qualify for Level 2 rates without submitting the total tax amount if your MCC is one of the following: 9399, 9211, 4784, 4111, 4131, 4215, 8211, 8220, 8398, 8661, 9222, 9311, 9402.</p>	<ol style="list-style-type: none"> 1. You must include at least one line item with amount, description, and quantity defined.

NOTE: Little & Co. always attempts to qualify your transactions for the optimal Interchange Rate. Although in some instances your transaction may qualify for either Level 2 or Level 3 rates without submitting all recommended fields, for the most consistent results, Little strongly recommends that you adhere to the guidelines detailed above.

Two child elements, `deliveryType` and `shippingAmount`, are required for Bill Me Later Authorization and Sale transactions.

Parent Elements:

[authorization](#), [capture](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#)

Child Elements: (all Optional)

[customerReference](#), [salesTax](#), [deliveryType](#), [taxExempt](#), [discountAmount](#), [shippingAmount](#), [dutyAmount](#), [shipFromPostalCode](#), [destinationPostalCode](#), [destinationCountryCode](#), [invoiceReferenceNumber](#), [orderDate](#), [detailTax](#), [lineItemData](#)

Example: enhancedData Structure

```
<enhancedData>
  <customerReference>Customer Reference</customerReference>
  <salesTax>Amount of Sales Tax Included in Transaction</salesTax>
  <deliveryType>TBD</deliveryType>
  <taxExempt>true or false</taxExempt>
  <discountAmount>Discount Amount Applied to Order</discountAmount>
  <shippingAmount>Amount to Transport Order</shippingAmount>
  <dutyAmount>Duty on Total Purchase Amount</dutyAmount>
  <shipFromPostalCode>Ship From Postal Code</shipFromPostalCode>
  <destinationPostalCode>Ship To Postal Code</destinationPostalCode>
  <destinationCountryCode>Ship To ISO Country Code</destinationCountryCode>
  <invoiceReferenceNumber>Merchant Invoice Number</invoiceReferenceNumber>
  <orderDate>Date Order Placed</orderDate>
  <detailTax>
    <taxIncludedInTotal>true or false</taxIncludedInTotal>
    <taxAmount>Additional Tax Amount</taxAmount>
    <taxRate>Tax Rate of This Tax Amount</taxRate>
    <taxTypeIdentifier>Tax Type Enum</taxTypeIdentifier>
    <cardAcceptorTaxId>Tax ID of Card Acceptor</cardAcceptorTaxId>
  </detailTax>
  <lineItemData>
    <itemSequenceNumber>Line Item Number within Order</itemSequenceNumber>
    <itemDescription>Description of Item</itemDescription>
    <productCode>Product Code of Item</productCode>
    <quantity>Quantity of Item</quantity>
    <unitOfMeasure>Unit of Measurement Code</unitOfMeasure>
    <taxAmount>Sales Tax or VAT of Item</taxAmount>
    <lineItemTotal>Total Amount of Line Item</lineItemTotal>
    <lineItemTotalWithTax>taxAmount + lineItemTotal</lineItemTotalWithTax>
    <itemDiscountAmount>Discount Amount</itemDiscountAmount>
    <commodityCode>Card Acceptor Commodity Code for Item</commodityCode>
    <unitCost>Price for One Unit of Item</unitCost>
  </lineItemData>
</enhancedData>
```

4.104 entryMode

The `entryMode` element is a required child of the `pos` element, which describes the method used for card data entry at the point of sale.

Type = String (Enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[pos](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
notused	terminal not used
keyed	card number manually entered
track1	track 1 read
track2	magnetic stripe read (track 2 when known or when the terminal makes no distinction between tracks 1 and 2.)
completeread	complete magnetic stripe read and transmitted

4.105 expDate

The `expDate` element is a child of the `card`, `token`, `paypage` elements, which specifies the expiration date of the card and is required for card-not-present transactions.

NOTE: Although the schema defines the `expDate` element as an optional child of the `card`, `token` and `paypage` elements, you must submit a value for card-not-present transactions.

Type = String; minLength = 4; maxLength = 4

Parent Elements:

[card](#), [newCardInfo](#), [newCardTokenInfo](#), [originalCard](#), [originalCardInfo](#), [originalCardTokenInfo](#), [originalToken](#), [paypage](#), [token](#), [updatedCard](#), [updatedToken](#)

Attributes:

None

Child Elements:

None

NOTE: You should submit whatever expiration date you have on file, regardless of whether or not it is expired/stale.
Little & Co. recommends all merchant with recurring and/or installment payments participate in the Automatic Account Updater program.

4.106 extendedCardResponse

The `extendedCardResponse` element is an optional child of the `accountUpdater` element, which contains two child elements, `code` and `message`. The codes/messages can be either “501 - The Account Was Closed.” or “504 - Contact the cardholder for updated information.”

Parent Elements:

[accountUpdater](#)

Attributes:

None

Child Elements:

[code](#), [message](#)

Example: newCardInfo Structure

```
<extendedCardResponse>
  <message>Message for Code</message>
  <code>Either 501 or 504</code>
</extendedCardResponse>
```

4.107 filtering

The `filtering` element is an optional child of either the Authorization or Sale request transaction. You use its child elements to selectively enable/disable the various Litle Card Filtering features. Setting either the `<international>` or `<chargeback>` child element to **false** disables that filtering feature for the transaction. The `<prepaid>` child can be set to **true** to enable the feature selectively, or set to **false** to disable the feature for the transaction, if you elected to use the **filter all** prepaid configuration option.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

Optional: [prepaid](#), [international](#), [chargeback](#)

Example: filtering Structure

```
<filtering>
  <prepaid>true or false</prepaid>
  <international>>false</international>
  <chargeback>>false</chargeback>
</filtering>
```


4.108 firstName

The `firstName` element is a child of the `billtoAddress` element, which specifies the first name of the account holder and is required for `echeckVerification` transactions.

NOTE: When performing an eCheck Verification for a corporate account, you must include values for the `firstName` and `lastName` elements. If you do not have the name of the check issuer, you can use a value of “unavailable” for both elements.

Type = String; minLength = N/A; maxLength = 25

Parent Elements:

[billToAddress](#)

Attributes:

None

Child Elements:

None

4.109 forceCapture

The `forceCapture` element is the parent element for all Force Capture transactions. These are specialized Capture transactions used when you do not have a valid Authorization for the order, but have fulfilled the order and wish to transfer funds. You can use this element in either Online or Batch transactions.

CAUTION: You must be authorized by Litle & Co. before processing this transaction type. In some instances, using a Force Capture transaction can lead to chargebacks and fines.

Parent Elements:

[litleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. This attribute is also used for Duplicate Transaction Detection. Failure to include a value for this attribute disables Duplicate Detection for the transaction. Please refer to Duplicate Transaction Detection on page 6 for additional information about the operation of Duplicate checking. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [amount](#), [orderSource](#), choice of [card](#), [token](#), or [paypage](#)

Optional: [billToAddress](#), [customBilling](#), [taxType](#), [enhancedData](#), [processingInstructions](#), [pos](#), [amexAggregatorData](#), [merchantData](#)

4.110 forceCaptureResponse

The `forceCaptureResponse` element is the parent element for information returned to you in response to a Force Capture transaction. It can be a child of either a `littleOnlineResponse` element or a `batchResponse` element.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the Force Capture transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the Force Capture transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the Force Capture transaction. minLength = 1 maxLength = 25
duplicate	Boolean	No	If the request is a duplicate (see Online Duplicate Checking on page 7), the response includes the duplicate flag set to true and the entire original response. Note: This attribute applies only to Online transaction responses.

Child Elements:

Required: [littleTxnId](#), [orderId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [tokenResponse](#), [accountUpdater](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.111 fraudResult

The `fraudResult` element is an optional child of the `authorizationResponse`, the `saleResponse` and the `authInformation` elements. It contains child elements defining any fraud checking results.

Parent Elements:

[authorizationResponse](#), [saleResponse](#), [authInformation](#)

Attributes:

None

Child Elements: (All Optional)

[authenticationResult](#), [avsResult](#), [cardValidationResult](#), [advancedAVSResults](#)

Example: fraudResult Structure

```
<fraudResult>
  <avsResult>00</avsResult>
  <cardValidationResult>N</cardValidationResult>
  <authenticationResult>2</authenticationResult>
  <advancedAVSResult>011</advancedAVSResult>
</fraudResult>
```

NOTE: The `<advancedAVSResults>` element applies only to American Express transactions. Also, you must be certified to use LittleXML version 7.3 or above and specifically enabled to use the Advanced AVS feature. Please consult your Little Relationship Manager for additional information.

4.112 fundingSource

The `fundingSource` element is an optional child of the `enhancedAuthResponse` element. Through its child elements, it provides information concerning whether the card used for the transaction is Prepaid and if so, the available balance.

Parent Elements:

[enhancedAuthResponse](#)

Attributes:

None

Child Elements:

Required: [type](#), [availableBalance](#), [reloadable](#), [prepaidCardType](#)

Example: fundingSource Structure

```
<fundingSource>
  <type>PREPAID</type>
  <availableBalance>0</availableBalance>
  <reloadable>YES|NO|UNKNOWN</reloadable>
  <prepaidCardType>GIFT</prepaidCardType>
</fundingSource>
```

NOTE: The `fundingSource` element and its child elements, `type` and `availableBalance` are associated with the Smart Authorization features (see [Smart Authorization Features](#) on page 11.) Please consult your Relationship Manager for additional information.

4.113 healthcareAmounts

The `healthcareAmount` element is a required child of the `healthcareIIAS` element. Through its child elements, it provides details about the dollar amount and type of IIAS qualified items purchased using Healthcare Prepaid cards.

The value used for the `totalHealthcareAmount` child must be the sum of the values applied to the following elements: `RxAmount`, `visionAmount`, `clinicOtherAmount`, and `dentalAmount`.

Parent Elements:

[healthcareIIAS](#)

Attributes:

None

Child Elements:

Required: [totalHealthcareAmount](#)

Optional: [RxAmount](#), [visionAmount](#), [clinicOtherAmount](#), [dentalAmount](#)

Example: fundingSource Structure

```
<healthcareAmounts>
  <totalHealthcareAmount>Total of Healthcare Items</totalHealthcareAmount>
  <RxAmount>Amount for Medications</RxAmount>
  <visionAmount>Amount for Vision Items</visionAmount>
  <clinicOtherAmount>Amount for Clinic Charges</clinicOtherAmount>
  <dentalAmount>Amount for Dental Charges</dentalAmount>
</healthcareAmounts>
```

4.114 healthcareIIAS

The `healthcareIIAS` element is an optional child of Authorization and Sale transactions. Through its child elements, it provides information about IIAS qualified items purchased using Healthcare Prepaid cards.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

Required: [healthcareAmounts](#), [IIASFlag](#)

Example: fundingSource Structure

```
<healthcareIIAS>
  <healthcareAmounts>
    <totalHealthcareAmount>Total of Healthcare Items</totalHealthcareAmount>
    <RxAmount>Amount for Medications</RxAmount>
    <visionAmount>Amount for Vision Items</visionAmount>
    <clinicOtherAmount>Amount for Clinic Charges</clinicOtherAmount>
    <dentalAmount>Amount for Dental Charges</dentalAmount>
  </healthcareAmounts>
  <IIASFlag>Y</IIASFlag>
</healthcareIIAS>
```

4.115 IIASFlag

The `IIASFlag` element is a required child of the `healthcareIIAS` element. This element only supports a value of **Y**.

Type = String (enum); **minLength** = N/A; **maxLength** = 1; **Valid Value** = Y

Parent Elements:

[healthcareIIAS](#)

Child Elements:

None

4.116 incomeAmount

The `incomeAmount` element is an optional child of the `customerInfo` element and defines the yearly income of the customer. It is used in combination with several other elements to provide information for some Bill Me Later transactions.

Type = Long; **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.117 incomeCurrency

The `incomeCurrency` element is an optional child of the `customerInfo` element and defines the currency of the `incomeAmount` element. The default value is USD (United States Dollars). It is used in combination with several other elements to provide information for some Bill Me Later transactions.

Type = String (Enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
AUD	Australian Dollar
CAD	Canadian Dollar
CHF	Swiss Francs
DKK	Denmark Kroner
EUR	Euro
GBP	United Kingdom Pound
HKD	Hong Kong Dollar
JPY	Japanese Yen
NOK	Norwegian Krone
NZD	New Zealand Dollar
SEK	Swedish Kronor
SGD	Singapore Dollar
USD (default)	United States Dollar

4.118 international

The `international` element is an optional child of the `filtering` element. To disable the filtering operation for a selected transaction include the `international` element with a setting of **false**.

Type = Boolean; **Valid Value** = false

Parent Elements:

[filtering](#)

Attributes:

None

Child Elements:

None

4.119 invoiceReferenceNumber

The `invoiceReferenceNumber` element is an optional child of the `enhancedData` element, which specifies the merchant's invoice number. If you do not know the invoice number, do not include this element.

Type = String; **minLength** = 1; **maxLength** = 15

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.120 issuerCountry

The `issuerCountry` element is an optional child of the `enhancedAuthResponse` element, which defines the country of the bank that issued the card submitted in the Authorization or Sale transaction.

Type = String; **minLength** = N/A; **maxLength** = 3

Parent Elements:

[enhancedAuthResponse](#)

Attributes:

None

Child Elements:

None

4.121 itemCategoryCode

The `itemCategoryCode` element is an optional child of the `billMeLaterRequest` element and defines the Bill Me Later item category for the type of product sold.

NOTE: According to the Bill Me Later documentation, merchants typically assign the applicable Item Category Code at the store or department level as opposed to the actual product level. The BML documentation goes on to state, “Your BML Implementation Project Manager will provide you the Item Category Codes that are best associated with your merchandise.”

For additional information, please refer to the Bill Me Later Implementation documentation.

Type = Integer; totalDigits = 4

Parent Elements:

[billMeLaterRequest](#)

Attributes:

None

Child Elements:

None

4.122 itemDescription

The `itemDescription` element is a required child of the `lineItemData` element, which provides a brief text description of the item purchased.

Type = String; **minLength** = N/A; **maxLength** = 26

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.123 itemDiscountAmount

The `itemDiscountAmount` element is an optional child of the `lineItemData` element, which specifies the item discount amount. Although an optional element, it is required by Visa for Level III Interchange rates. The value must be greater than or equal to 0. The decimal is implied.

Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.124 itemSequenceNumber

The `itemSequenceNumber` element is an optional child of the `lineItemData` element (required for Visa transactions). When providing line item data, you must number each item sequentially starting with 1.

Type = Integer; **minInclusive value** = 1, **maxInclusive value** = 99

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.125 lastName

The `lastName` element is a child of the `billtoAddress` element, which specifies the last name of the account holder and is required for `echeckVerification` transactions.

NOTE: When performing an eCheck Verification for a corporate account, you must include values for the `firstName` and `lastName` element. If you do not have the name of the check issuer, you can use a value of “unavailable” for both elements.

Type = String; minLength = N/A; maxLength = 25

Parent Elements:

[billToAddress](#)

Attributes:

None

Child Elements:

None

4.126 lineItemData

The `lineItemData` element contains several child elements used to define information concerning individual items in the order. Although the schema defines it as an optional child of the `enhancedData` element, it is required for Level III interchange rates.

NOTE: MasterCard and Visa allow up to 99 instances of this element in a transaction. American Express allows a maximum of 4 instances of this element in a transaction.

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

Required: [itemDescription](#)

Optional: [itemSequenceNumber](#), [productCode](#), [quantity](#), [unitOfMeasure](#), [taxAmount](#), [lineItemTotal](#), [lineItemTotalWithTax](#), [itemDiscountAmount](#), [commodityCode](#), [unitCost](#), [detailTax](#)

NOTE: When including the `lineItemData` element, please be aware of the following rules for its child elements:

- `itemSequenceNumber` is required by Visa
- `productCode`, `quantity`, `unitOfMeasure`, and `lineItemTotal` are required by Visa and MasterCard
- `itemDiscountAmount`, `commodityCode`, and `unitCost` are required by Visa for Level III Interchange rates

Example: lineItemData Structure

```
<lineItemData>
  <itemSequenceNumber>Line Item Number within Order</itemSequenceNumber>
  <itemDescription>Description of Item</itemDescription>
  <productCode>Product Code of Item</productCode>
  <quantity>Quantity of Item</quantity>
  <unitOfMeasure>Unit of Measurement Code</unitOfMeasure>
  <taxAmount>Sales Tax or VAT of Item</taxAmount>
```

```
<lineItemTotal>Total Amount of Line Item</lineItemTotal>
<lineItemTotalWithTax>taxAmount + lineItemTotal</lineItemTotalWithTax>
<itemDiscountAmount>Discount Amount</itemDiscountAmount>
<commodityCode>Card Acceptor Commodity Code for Item</commodityCode>
<unitCost>Price for One Unit of Item</unitCost>
<detailTax>
  <taxIncludedInTotal>true or false</taxIncludedInTotal>
  <taxAmount>Additional Tax Amount</taxAmount>
  <taxRate>Tax Rate of This Tax Amount</taxRate>
  <taxTypeIdentifier>Tax Type Enum</taxTypeIdentifier>
  <cardAcceptorTaxId>Tax ID of Card Acceptor</cardAcceptorTaxId>
</detailTax>
</lineItemData>
```

4.127 lineItemTotal

The `lineItemTotal` element is an optional child of the `lineItemData` element, which specifies the total cost of the line items purchased, not including tax. For example, if the order was for 500 pencils at \$1.00 each, the `lineItemTotal` would be \$500. Although an optional element, it is required by Visa and MasterCard when specifying line item data. The decimal is implied. Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.128 lineItemTotalWithTax

The `lineItemTotalWithTax` element is an optional child of the `lineItemData` element, which specifies the total cost of the line items purchased including tax. If the tax is not known, do not include this element. The decimal is implied. Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.129 litleOnlineRequest

This is the root element for all LittleXML Online requests.

Parent Elements:

None

Attributes:

Attribute Name	Type	Required?	Description
version	String	Yes	Defines the LittleXML schema version against which the XML is validated. The current version is 8.10, but you may be using an older version. minLength = N/A maxLength = 10
xmlns	String	Yes	Defines the URI of the schema definition. This is a fixed location and must be specified as: http://www.litle.com/schema . minLength = N/A maxLength = 38
merchantId	String	Yes	A unique string used to identify the merchant within the Litle system. minLength = N/A maxLength = 50 Note: International currencies are supported on a per merchantId basis.

Child Elements:

Required: [authentication](#)

One of the following required: [authorization](#), [authReversal](#), [capture](#), [echeckVoid](#), [forceCapture](#), [captureGivenAuth](#), [credit](#), [echeckCredit](#), [echeckRedeposit](#), [echeckSale](#), [echeckVerification](#), [echeckVoid](#), [registerTokenRequest](#), [sale](#), [void](#)

4.130 littleOnlineResponse

This is the root element for all LittleXML Online responses.

Parent Elements:

None

Attributes:

Attribute Name	Type	Required?	Description
version	String	Yes	Defines the LittleXML schema version against which the XML is validated. The current version is 8.10, but you may be using an older version. minLength = N/A maxLength = 10
xmlns	String	Yes	Defines the URI of the schema definition. This is a fixed location and must be specified as: http://www.little.com/schema . minLength = N/A maxLength = 38
response	String	Yes	Indicates whether your XML syntax passed validation. Expected values are as follows: 0 - XML validation succeeded. 1 - XML validation failed. See the message attribute for more details. minLength = N/A maxLength = 3
message	String	Yes	XML validation error message. Expected values are as follows: <ul style="list-style-type: none">If the response attribute returns 0, the message attribute returns the text "Valid Format."If the response attribute returns 1, the message attribute returns an error message that helps you to identify and troubleshoot the syntax problem. See XML Validation Error Messages on page 454 for example messages. minLength = N/A maxLength = 512

Child Elements:

One of the following required: [authorizationResponse](#), [authReversalResponse](#), [captureResponse](#), [forceCaptureResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [echeckCreditResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [registerTokenResponse](#), [saleResponse](#), [voidResponse](#)

4.131 littleRequest

This is the root element for all LittleXML Batch requests.

Parent Elements:

None

Attributes:

Attribute Name	Type	Required?	Description
version	String	Yes	Defines the LittleXML schema version against which the XML is validated. The current version is 8.10, but you may be using an older version. minLength = N/A maxLength = 10
xmlns	String	Yes	Defines the URI of the schema definition. This is a fixed location and must be specified as: <code>http://www.little.com/schema</code> . minLength = N/A maxLength = 38
id	String	No	A unique string to identify the session within the Little system. minLength = N/A maxLength = 25
numBatchRequests	Integer	Yes	Defines the total number of batchRequest children included in the <code>littleRequest</code> . If the <code>littleRequest</code> contains only an <code>RFRRequest</code> , then set this attribute to "0".

Child Elements:

Required: [authentication](#)

One of the following required: [batchRequest](#), [RFRRequest](#)

4.132 litleResponse

This is the root element for all LitleXML Batch responses.

Parent Elements:

None

Attributes:

Attribute Name	Type	Required?	Description
version	String	Yes	Defines the LitleXML schema version against which the XML is validated. The current version is 8.10, but you may be using an older version. minLength = N/A maxLength = 10
xmlns	String	Yes	Defines the URI of the schema definition. This is a fixed location and must be specified as: http://www.litle.com/schema . minLength = N/A maxLength = 38
id	String	No	The response returns the same value submitted in the authorization transaction. minLength = N/A maxLength = 25
response	String	Yes	Indicates whether your XML syntax passed validation. Expected values are as follows: 0 - XML validation succeeded. 1 - XML validation failed. See the message attribute for more details. minLength = N/A maxLength = 3
message	String	Yes	XML validation error message. Expected values are as follows: <ul style="list-style-type: none">If the response attribute returns 0, the message attribute returns the text "Valid Format."If the response attribute returns 1, the message attribute returns an error message that helps you to identify and troubleshoot the syntax problem. See XML Validation Error Messages on page 454 for example messages. minLength = N/A maxLength = 512
litleSessionId	Long	Yes	A unique value assigned by Litle to identify the session. minLength = N/A maxLength = 19

Child Elements:

One of the following required: [batchResponse](#), [RFRResponse](#)

4.133 littleSessionId

The `littleSessionId` element is a child of the `RFRRequest` element used to request the response from a previously submitted Batch. The value of the `littleSessionId` must be the same as the value returned in the corresponding attribute of the `littleResponse`.

Type = Long; **minLength** = N/A; **maxLength** = 19

Parent Elements:

[RFRRequest](#)

Attributes:

None

Child Elements:

None

4.134 littleToken

The `littleToken` element defines the value of the token. The system returns this value in XML responses when issuing new tokens to replace account numbers. The length of the token is the same as the length of the submitted account number for credit card tokens or a fixed length of seventeen (17) characters for eCheck account tokens.

Type = String; **minLength** = 13; **maxLength** = 25

Parent Elements:

The `littleToken` element is an optional child of each listed parent element.

[registerTokenResponse](#), [tokenResponse](#), [newCardTokenInfo](#), [originalCardTokenInfo](#), [originalToken](#), [originalTokenInfo](#), [newTokenInfo](#), [updatedToken](#), [token](#), [echeckToken](#)

Attributes:

None

Child Elements:

None

4.135 littleTxnId

The `littleTxnId` element is used to identify transactions in the Little system. The system returns this element in XML responses. You use it in various requests to reference the original transaction. For example, when you submit a Capture transaction, you include the `littleTxnId` for the associated Authorization.

Type = Long; **minLength** = N/A; **maxLength** = 19

Parent Elements:

This element is a required child of the following: [accountUpdateResponse](#), [authorizationResponse](#), [authReversalResponse](#), [capture](#), [captureResponse](#), [credit](#), [creditResponse](#), [captureGivenAuthResponse](#), [echeckCredit](#), [echeckCreditResponse](#), [echeckRedeposit](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [echeckVoid](#), [echeckVoidResponse](#), [forceCapture](#), [forceCaptureResponse](#), [saleResponse](#), [void](#), [voidResponse](#)

NOTE:	Although the schema shows the <code>littleTxnId</code> element as an optional child of the <code>authorization</code> , <code>echeckSale</code> , <code>echeckVerification</code> , and <code>sale</code> transactions, under normal circumstances, merchants would never use this option.
--------------	--

Attributes:

None

Child Elements:

None

4.136 merchantData

The `merchantData` element is an optional child element of several transaction types. You can use its children to track transactions based upon marketing campaigns, affiliates, or other user defined parameter.

NOTE: At present this data is not included in any reports. In the future it will be added to the Session export and the Net Settled Sales export reports.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [echeckCredit](#), [echeckSale](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements (all optional):

[affiliate](#), [campaign](#), [merchantGroupingId](#)

4.137 merchantGroupingId

The `merchantGroupingId` element is an optional child element of the `merchantData` element. You can use it to track transactions based upon this user defined parameter.

NOTE: At present this data is not included in any reports. In the future it will be added to the Session export and the Net Settled Sales export reports.

Type = String; minLength = N/A; maxLength = 25

Parent Elements:

[merchantData](#)

Attributes:

None

Child Elements:

None

4.138 merchantId

The `merchantId` element is a child of the `accountUpdateFileRequestData` element used when you request an Account Update file. This value is a unique string used to identify the merchant within the Little system.

Type = String; **minLength** = N/A; **maxLength** = 50

Parent Elements:

[accountUpdateFileRequestData](#)

Attributes:

None

Child Elements:

None

NOTE:	Several elements use <code>merchantId</code> as an attribute, including <code>batchRequest</code> , <code>batchResponse</code> , and <code>littleOnlineRequest</code> .
--------------	---

4.139 message

The `message` element contains a brief definition of the response code returned for the transaction.

When it is a child of the `extendedCardResponse` element, the only values allowed are either "The account was closed," or "Contact the cardholder for updated information."

For a complete list of Litle & Co. response codes and associated messages, please refer to [Appendix A](#).

Type = String; **minLength** = N/A; **maxLength** = 512

Parent Elements:

[authorizationResponse](#), [captureResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [echeckCreditResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [echeckVoidResponse](#), [extendedCardResponse](#), [forceCaptureResponse](#), [saleResponse](#), [voidResponse](#)

Attributes:

None

Child Elements:

None

4.140 middleInitial

The `middleInitial` element is a child of the `billtoAddress` element, which specifies the middle initial of the account holder. It is an optional element used for `echeckVerification` transactions.

Type = String; **minLength** = N/A; **maxLength** = 1

Parent Elements:

[billToAddress](#)

Attributes:

None

Child Elements:

None

4.141 name

The `name` element defines the customer name in both the `billToAddress` and `shipToAddress` elements.

Type = String; **minLength** = N/A; **maxLength** = 100

Parent Elements:

[billToAddress](#), [shipFromPostalCode](#)

NOTE: The `name` element is required for Echeck transactions. If you do not submit the customer name in an Echeck transaction, Litle returns Response Code 330 - Invalid Payment Type.

Attributes:

None

Child Elements:

None

4.142 newAccountInfo

The `newAccountInfo` element is an optional child of the `accountUpdater` element, which contains child elements providing the updated information for the submitted account.

Parent Elements:

[accountUpdater](#)

Attributes:

None

Child Elements:

[accType](#), [accNum](#), [routingNum](#)

Example: newAccountInfo Structure

```
<newAccountInfo>
  <accType>Account Type</accType>
  <accNum>New Account Number</accNum>
  <routingNum>New Routing Number</routingNum>
</newAccountInfo>
```

4.143 newCardInfo

The `newCardInfo` element is an optional child of the `accountUpdater` element, which contains child elements providing the updated information for the submitted card.

Parent Elements:

[accountUpdater](#)

Attributes:

None

Child Elements:

[type](#), [number](#), [expDate](#)

Example: newCardInfo Structure

```
<newCardInfo>
  <type>Card Type</type>
  <number>New Account Number</number>
  <expDate>New Expiration Date</expDate>
</newCardInfo>
```

4.144 newCardTokenInfo

The `newCardTokenInfo` element is an optional child of the `accountUpdater` element, which contains child elements providing the updated token information for the submitted token.

Parent Elements:

[accountUpdater](#)

Attributes:

None

Child Elements:

[littleToken](#), [type](#), [expDate](#), [bin](#)

Example: newCardInfo Structure

```
<newCardTokenInfo>
  <littleToken>New Token</littleToken>
  <type>Card Type</type>
  <expDate>New Expiration Date</expDate>
  <bin>New Card BIN</bin>
</newCardTokenInfo>
```

4.145 newTokenInfo

The `newTokenInfo` element is an optional child of the `accountUpdater` element, which contains child elements providing the updated information for the submitted account. The system returns this information when processing a tokenized eCheck transactions and a change (NOC) is found against the account.

Parent Elements:

[accountUpdater](#)

Attributes:

None

Child Elements:

[accType](#), [littleToken](#), [routingNum](#)

Example: newAccountInfo Structure

```
<newTokenInfo>
  <accType>Account Type</accType>
  <littleToken>New Token Number</littleToken>
  <routingNum>New Routing Number</routingNum>
</newTokenInfo>
```

4.146 nextRecycleTime

The `nextRecycleTime` element is an optional child of the `recycleAdvice` element, which specifies the date and time (in GMT) recommended for the next recycle of the declined Authorization/Sale transaction. The format of the element is YYYY-MM-DDTHH:MM:SSZ. For example, 2011-04-21T11:00:00Z.

NOTE: Per the ISO8601 standard, the Z appended to the end of the date/time stamp indicates the time is GMT.

Type = dateTime; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[recycleAdvice](#)

Attributes:

None

Child Elements:

None

4.147 number

The `number` element is an optional child of the `accountInformation` element (used in Authorization and Sale responses) and defines the account number associated with the transaction.

Type = String; **minLength** = 13; **maxLength** = 25

Parent Elements:

[accountInformation](#), [card](#), [newCardInfo](#), [originalCardInfo](#)

Attributes:

None

Child Elements:

None

4.148 orderDate

The `orderDate` element is an optional child of the `enhancedData` element, which specifies the date the order was placed. If you do not know the order date, do not include this element.

Type = Date; **Format** = YYYY-MM-DD

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.149 orderId

The orderId element defines a merchant-assigned value representing the order in the merchant's system.

Type = String; **minLength** = N/A; **maxLength** = 25

Parent Elements:

[accountUpdate](#), [accountUpdateResponse](#), [authorization](#), [authorizationResponse](#), [captureResponse](#) (Batch only), [credit](#), [creditResponse](#), [captureGivenAuth](#), [captureGivenAuthResponse](#), [echeckCredit](#), [echeckCreditResponse](#), [echeckSale](#), [echeckSalesResponse](#), [echeckVerification](#), [echeckVerificationResponse](#), [forceCapture](#), [forceCaptureResponse](#), [sale](#), [saleResponse](#), [registerTokenRequest](#)

Attributes:

None

Child Elements:

None

4.150 orderSource

The orderSource element defines the order entry source for the type of transaction.

Type = Choice (enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[authorization](#), [credit](#), [captureGivenAuth](#), [echeckCredit](#), [echeckSale](#), [echeckVerification](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements:

None

Enumerations:

NOTE: If you submit the wrong orderSource value, Little returns the response code 370 - Internal System Error - Contact Little. Also, Bill Me Later transactions must use an orderSource of either ecommerce, mailorder, or telephone. Use of other types will cause the authorization to fail.

Enumeration	Description
3dsAuthenticated	The transaction qualified as CPS/e-Commerce Preferred as an authenticated purchase. Use this value only if you authenticated the cardholder.
3dsAttempted	The transaction qualified as CPS/e-Commerce Preferred as an attempted authentication. Use this value only if you attempted to authenticate the cardholder and either the Issuer or cardholder is not participating in Verified by Visa.
ecommerce	The transaction is an Internet or electronic commerce transaction.
installment	The transaction in an installment payment.
mailorder	The transaction is for a single mail order transaction.
recurring	The transaction is a recurring transaction.
retail	The transaction is a Swiped or Keyed Entered retail purchase transaction.
telephone	The transaction is for a single telephone order.
recurringtel	(eCheck only) The transaction is a recurring eCheck transaction initiated via telephone

4.151 originalAccountInfo

The `originalAccountInfo` element is an optional child of the `accountUpdater` element, which contains child elements providing the original information for the submitted account.

Parent Elements:

[accountUpdater](#)

Attributes:

None

Child Elements:

[accType](#), [accNum](#), [routingNum](#)

Example: originalAccountInfo Structure

```
<originalAccountInfo>
  <accType>Account Type</accType>
  <littleToken>Original Token Number</littleToken>
  <routingNum>Original Routing Number</routingNum>
</originalAccountInfo>
```

4.152 originalCard

The `originalCard` element is an optional child of the `accountUpdateResponse` element, which contains child elements providing the original information for the submitted card.

Parent Elements:

[accountUpdateResponse](#)

Attributes:

None

Child Elements:

[type](#), [number](#), [expDate](#)

Example: originalCard Structure

```
<originalCard>
  <type>Card Type</type>
  <number>Old Account Number</number>
  <expDate>Old Expiration Date</expDate>
</originalCard>
```

4.153 originalCardInfo

The `originalCardInfo` element is an optional child of the `accountUpdater` element, which contains child elements providing the original information for the submitted card.

Parent Elements:

[accountUpdater](#)

Attributes:

None

Child Elements:

[type](#), [number](#), [expDate](#)

Example: originalCard Structure

```
<originalCardInfo>
  <type>Card Type</type>
  <number>Old Account Number</number>
  <expDate>Old Expiration Date</expDate>
</originalCardInfo>
```

4.154 originalCardTokenInfo

The `originalCardTokenInfo` element is an optional child of the `accountUpdater` element, which contains child elements providing the original information for the submitted token.

Parent Elements:

[accountUpdater](#)

Attributes:

None

Child Elements:

[littleToken](#), [type](#), [expDate](#), [bin](#)

Example: originalCard Structure

```
<originalCardTokenInfo>
  <littleToken>Old Token</littleToken>
  <type>Card Type</type>
  <expDate>Old Expiration Date</expDate>
  <bin>Old Card BIN</bin>
</originalCardTokenInfo>
```


4.155 originalToken

The `originalToken` element is an optional child of the `accountUpdateResponse` element, which contains child elements providing the original information for the submitted token.

Parent Elements:

[accountUpdateResponse](#)

Attributes:

None

Child Elements:

[type](#), [number](#), [expDate](#), [bin](#)

Example: originalCard Structure

```
<originalToken>
  <littleToken>Old Token Number</littleToken>
  <expDate>Old Expiration Date</expDate>
  <type>Card Type</type>
  <bin>Card BIN</bin>
</originalToken>
```

4.156 originalTokenInfo

The `originalTokenInfo` element is an optional child of the `accountUpdater` element, which contains child elements providing the original token information for the submitted account. The system returns this information when processing a tokenized eCheck transactions and a change (NOC) is found against the account.

Parent Elements:

[accountUpdater](#)

Attributes:

None

Child Elements:

[accType](#), [littleToken](#), [routingNum](#)

Example: originalAccountInfo Structure

```
<originalTokenInfo>
  <accType>Account Type</accType>
  <littletoken>Old Account Number</littletoken>
  <routingNum>Old Routing Number</routingNum>
</originalTokenInfo>
```

4.157 password

The `password` element is a required child of the `authentication` element. It is used in combination with the `user` element to authenticate that the message is from a valid source.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[authentication](#)

Attributes:

None

Child Elements:

None

4.158 payerId

The `payerId` element is a required child of the `paypal` element for all cases except for an Online Credit transaction, where you can choose between this element and the `payerEmail` element. This element specifies the Payer Id returned from PayPal.

NOTE: The value of the Little & Co. `<payerId>` element must match the **PAYERID** value returned by the **GetExpressCheckout** call operation to PayPal.

Type = String; minLength = 1; maxLength = 17

Parent Elements:

[paypal](#)

Attributes:

None

Child Elements:

None

4.159 paypage

The `paypage` element defines Pay Page account information. It replaces the `card` or `token` elements in transactions using the Pay Page feature of the Little Vault solution. When you submit the `paypage` element in a request, response messages will include token information.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements:

Required: [paypageRegistrationId](#)

Optional: [expDate](#), [cardValidationNum](#), [type](#)

NOTE: Although the schema defines the `expDate` element as an optional child of the `paypage` element, you must submit a value for card-not-present transactions.

Example: Example: paypage Structure

```
<paypage>
  <paypageRegistrationId>Registration ID from PayPage</paypageRegistrationId>
  <expDate>Expiration Date</expDate>
  <cardValidationNum>Card Validation Number</cardValidationNum>
  <type>Method of Payment</type>
</paypage>
```

4.160 paypageRegistrationId

The `paypageRegistrationId` element is a required child of the `paypage` element, and specifies the Pay Page Registration ID generated by `securepaypage.little.com` (Little Pay Page). It can also be used in a Register Token Request to obtain a token based on Pay Page activity prior to submitting an Authorization or Sale transaction.

Type = String; **minLength** = N/A; **maxLength** = 512

Parent Elements:

[paypage](#), [registerTokenRequest](#)

Attributes:

None

Child Elements:

None

4.161 paypal

The `paypal` element defines paypal account information. It replaces the `card` or `token` elements in transactions using PayPal as a payment method.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

Required: [payerId](#), [transactionId](#)

Optional: [token](#)

Example: paypal Structure

```
<paypal>
  <payerId>PayPal Customer Identifier</payerId>
  <token>Token Value Returned</token>
  <transactionId>PayPal Transaction ID</transactionId>
</paypal>
```

4.162 paypalOrderComplete

The `paypalOrderComplete` element is an optional child of both the `capture` and `sale` elements, but is required to close a PayPal order. Set the value to **true** to close the order, when you have fulfilled the order and do not need to send any further auths or deposits against it. Set the value to **false** to keep the order open for additional auths or deposits.

Type = Boolean; **Valid values** = true or false

Parent Elements:

[capture](#), [sale](#)

Attributes:

None

Child Elements:

None

4.163 phone

The `phone` element has two different uses in LittleXML depending upon the parent element. When used as a child of either the `billToAddress` or `shipToAddress` elements, it defines the customers phone number. When used as a child of the `customBilling` element, it defines the phone number of the merchant.

4.163.0.1 phone as a child of `billToAddress` and `shipToAddress`

The `phone` element defines the customer's phone number in both the `billToAddress` and `shipToAddress` elements.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[billToAddress](#), [shipFromPostalCode](#)

Attributes:

None

Child Elements:

None

4.163.0.2 phone as a child of `customBilling`

The `phone` element defines the merchant's phone number. The string can only contain numbers (0 through 9). Letters and special characters are not allowed.

Type = String; **minLength** = N/A; **maxLength** = 13

Parent Elements:

[customBilling](#)

Attributes:

None

Child Elements:

None

4.164 pos

The `pos` element contains child elements used to specify information required when submitting authorization, `captureGivenAuth`, `credit`, `forceCapture`, and `sale` transactions from point of sale terminals.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements:

[capability](#), [entryMode](#), [cardholderId](#)

Example: pos Structure

```
<pos>
  <capability>Capabilty Enumeration</capability>
  <entryMode>Entry Mode Enumeration</entryMode>
  <cardholderId>Cardholder ID Enumeration</cardholderId>
</pos>
```

4.165 postDate

The `postDate` element defines the date the transaction was posted. The format is YYYY-MM-DD. It occurs only in response to Online transactions.

NOTE: Although the schema defines this element as optional in all cases except for the `voidResponse` parent element, the system returns it in the response for all Online transactions.

Type = Date; minLength = N/A; maxLength = 10

Parent Elements:

[authorizationResponse](#), [authReversalResponse](#), [captureResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [echeckCreditResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [forceCaptureResponse](#), [saleResponse](#), [voidResponse](#)

Attributes:

None

Child Elements:

None

4.166 postDay

The `postDay` element is an optional child of the `accountUpdateFileRequestData` element that defines the date you submitted the Account Updater request. The format is YYYY-MM-DD.

NOTE: This is also the same date that Little & Co. created the Account Updater acknowledgement file.

Type = Date; minLength = N/A; maxLength = 10

Parent Elements:

[accountUpdateFileRequestData](#)

Attributes:

None

Child Elements:

None

4.167 preapprovalNumber

The `preapprovalNumber` element is an optional child of the `billMeLaterRequest` element, which you use to specify the pre-approval number issued by Bill Me Later. If you include this element, the value must be 16 digits in length. Do not include this element to indicate there is no pre-approval. Internal pre-approval is indicated by using 1 as the first digit.

Type = String; **minLength** = 13; **maxLength** = 25

Parent Elements:

[billMeLaterRequest](#)

Attributes:

None

Child Elements:

None

4.168 prepaid

The `prepaid` element is an optional child of the `filtering` element. How you choose to implement the Prepaid Filtering feature determines the use of the `prepaid` element. If your configuration filters all prepaid card transactions, you can disable the feature on selected transactions by including the `prepaid` element with a setting of **false**. If your configuration filters prepaid card transactions on a per transaction basis, you enable the filtering on a selected transaction by including the `prepaid` element with a setting of **true**.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[filtering](#)

Attributes:

None

Child Elements:

None

4.169 prepaidCardType

The `prepaidCardType` element is an optional child of the `enhancedAuthResponse` element, which specifies the type of prepaid card submitted in the Authorization or Sale transaction. For example, a few of the possible values are: GIFT, PAYROLL, and GENERAL_PREPAID

Type = String; **minLength** = N/A; **maxLength** = 50

Parent Elements:

[fundingSource](#)

Attributes:

None

Child Elements:

None

4.170 processingInstructions

The `processingInstructions` element contains a child element that allows you to specify whether or not the system performs velocity checking on the transaction.

Parent Elements: (optional for all)

[authorization](#), [capture](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#), [void](#)

Attributes:

None

Child Elements:

[bypassVelocityCheck](#)

NOTE:	Please consult your Relationship Manager for additional information concerning Velocity Checking.
--------------	--

Example: processingInstructions Structure

```
<processingInstructions>
  <bypassVelocityCheck>true or false</bypassVelocityCheck>
</processingInstructions>
```


4.171 productCode

The `productCode` element is an optional child of the `lineItemData` element, which specifies the product code of the purchased item. Although an optional element, it is required by Visa and MasterCard when specifying line item data.

Type = String; **minLength** = 1; **maxLength** = 12

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.172 quantity

The `quantity` element is an optional child of the `lineItemData` element, which specifies the number of items purchased. Although an optional element, it is required by Visa and MasterCard when specifying line item data. The value must be greater than zero, but no more than 12 digits not including the decimal point.

Type = Decimal; **minInclusive** = 0; **totalDigits** = 12

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.173 recycleAdvice

The `recyclingAdvice` element contains a two child elements that either specifies the date and time (in GMT) recommended for the next recycle of the declined Authorization/Sale transaction or indicates that there is no additional recycling advice. The two children are mutually exclusive.

Parent Elements: (optional for all)

[recycling](#)

Attributes:

None

Child Elements:

[nextRecycleTime](#), [recycleAdviceEnd](#)

NOTE: The `recycleAdvice` element contains either a `nextRecycleTime` or `recycleAdviceEnd` element, but not both.

Example: recycleAdvice Structure - with recommended Date:Time

```
<recycleAdvice>
  <nextRecycleTime>2011-04-15T12:00:00</nextRecycleTime>
</recycleAdvice>
```

Example: recycleAdvice Structure - with end message

```
<recycleAdvice>
  <recycleAdviceEnd>End of Advice</recycleAdviceEnd>
</recycleAdvice>
```

4.174 recycleAdviceEnd

The `recycleAdviceEnd` element is an optional child of the `recycleAdvice` element and signifies that no further recycling recommendations are available.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[recycleAdvice](#)

Attributes:

None

Child Elements:

None

4.175 recycleBy

The `recycleBy` element is a required child of the `recyclingRequest` element and determines the use of the Little Recycling Engine/Recycling Advice. The default setting is `Little`, so omitting this element is the same as submitting a value of `Little`.

NOTE: Also, if your Little Merchant Profile includes a preset percentage split of transactions between merchant and Little controlled, then settings of `Merchant` and `Little` are ignored; you can still use a a setting of `None` to exclude the transaction.

Also, although the default setting is normally `Little`, it can be altered in your merchant profile to a setting of `Merchant`.

Type = String (Enum); minLength = N/A; maxLength = N/A

Parent Elements:

[recyclingRequest](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enum	Description
Merchant	This setting indicates that the merchant controls the recycling of the transaction. For A/B comparison testing, transactions using this setting will be counted as merchant controlled. After setting this value in the initial transaction, subsequent transactions should have same value. Any different value will be ignored.
Little	This setting indicates either that the Little Recycling Engine controls the recycling of the transaction or the recycling of the transaction will follow the Recycling Advice returned in the response message. For A/B comparison testing, transactions using this setting will be counted as Little controlled. After setting this value in the initial transaction, subsequent transactions should have same value. Any different value will be ignored.
None	For A/B comparison testing, transactions using this setting are excluded from all counts. These transactions will not be counted as either merchant or Little controlled.

4.176 recycling

The `recycling` element contains a child element that specify either the recommended date and time for the next recycling attempt for the declined Authorization/Sale transaction or a statement that no further advice is available.

Parent Elements:

[authorizationResponse](#), [saleResponse](#)

Attributes:

None

Child Elements:

[nextRecycleTime](#), [recycleAdviceEnd](#)

NOTE: The `recycleAdvice` element contains either a `nextRecycleTime` or `recycleAdviceEnd` element, but not both.

Example: recycling Structure - with recommended Date:Time

```
<recycling>
  <recycleAdvice>
    <nextRecycleTime>2011-04-15T12:00:00</nextRecycleTime>
  </recycleAdvice>
</recycling>
```

Example: recycling Structure - with end message

```
<recycling>
  <recycleAdvice>
    <recycleAdviceEnd>End of Advice</recycleAdviceEnd>
  </recycleAdvice>
</recycling>
```

4.177 recyclingRequest

The `recyclingrequest` element is an optional child of the `authorization` and `sale` transactions, which contains a child element that specifies who is responsible for recycling the transaction. This element only applies to merchants using either the Little Recycling Engine or Recycling Advice.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

[recycleBy](#)

Example: recyclingRequest Structure

```
<recyclingRequest>
  <recycleBy>Merchant or Little or None</recycleBy>
</recyclingRequest>
```

4.178 registerTokenRequest

The `registerTokenRequest` element is the parent element for the Register Token transaction. You use this transaction type when you wish to submit an account number for tokenization, but there is no associated payment transaction.

You can use this element in either Online or Batch transactions.

NOTE: When submitting `registerTokenRequest` elements in a `batchRequest`, you must also include a `numTokenRegistrations=` attribute in the `batchRequest` element.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute defining the merchant sub-group in the user interface where this transaction displays. Also see Coding for Report Groups on page 9 for information. minLength = 1 maxLength = 25

Child Elements:

Required: either [accountNumber](#), [echeckForToken](#), or [paypageRegistrationId](#)

Optional: [orderId](#)

4.179 registerTokenResponse

The `registerTokenResponse` element is the parent element for the Little response to `registerTokenRequest` transactions. You receive this transaction type in response to the submission of an account number for tokenization in a `registerTokenRequest` transaction.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the <code>registerTokenRequest</code> transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the <code>registerTokenRequest</code> transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the <code>registerTokenRequest</code> transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [response](#), [message](#), [responseTime](#)

Optional: [eCheckAccountSuffix](#), [orderId](#), [littleToken](#), [bin](#), [type](#)

4.180 reloadable

The `reloadable` element is an optional child of the `fundingSource` element and defines whether the prepaid card is reloadable.

Type = String (Enum); **Enumerations** = YES, NO, or UNKNOWN

Parent Elements:

[fundingSource](#)

Attributes:

None

Child Elements:

None

4.181 residenceStatus

The `residenceStatus` element is an optional child of the `customerInfo` element and defines the type of domicile in which the customer resides. It is used in combination with several other elements to provide required information for some Bill Me Later transactions.

Type = String (Enum); **Enumerations** = Own, Rent, or Other

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.182 response

The `response` element contains a three digit numeric code which specifies either that the transaction is approved (000 code) or declined. The `message` element provides a brief definition of the response code.

For a complete list of Little & Co. response codes and associated messages, please refer to Appendix A.

Type = String; **minLength** = N/A; **maxLength** = 3

Parent Elements:

[authorizationResponse](#), [authReversalResponse](#), [captureResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [echeckCreditResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [echeckVoidResponse](#), [forceCaptureResponse](#), [registerTokenResponse](#) [saleResponse](#), [voidResponse](#)

Attributes:

None

Child Elements:

None

4.183 responseTime

The `responseTime` element provides a date/time stamp of the response. The format of the element is YYYY-MM-DDTHH:MM:SS. For example, 2009-12-21T11:37:04.

Type = dateTime; **minLength** = N/A; **maxLength** = 19

Parent Elements:

[authorizationResponse](#), [authReversalResponse](#), [captureResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [echeckCreditResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [echeckVoidResponse](#), [forceCaptureResponse](#), [registerTokenResponse](#) [saleResponse](#), [voidResponse](#)

Attributes:

None

Child Elements:

None

4.184 RFRRequest

The `RFRRequest` element is an optional child of a `littleRequest` element. You can use this type of request in one of two ways.

- To request a session response from a previously processed `littleRequest`, include the `littleSessionId` child. The resulting RFR response will duplicate the original session response (Authorization, Credit, Capture, or Sale response) associated with the `littleSessionId`. The session ID returned in the response will be the session ID of the original session.
- To request an Account Updater completion response file, include the `accountUpdateFileRequestData` element. If the completion file is ready, it is returned. If the completion file is not ready, you receive an `RFRResponse` message with the response attribute set to 1 and the message attribute reading, “The account Update file is not ready yet. Please try again later.”

Parent Elements:

[littleRequest](#)

Attributes:

None

Child Elements: (Choice of)

[littleSessionId](#) or [accountUpdateFileRequestData](#)

Example: RFRRequest Structure - Batch

```
<RFRRequest>
  <littleSessionId>Session ID</littleSessionId>
</RFRRequest>
```

Example: RFRRequest Structure - Account Updater

```
<RFRRequest>
  <accountUpdateFileRequestData>
    <merchantId>Merchant ID</merchantId>
    <postDay>Post Date</postDay>
  </accountUpdateFileRequestData>
</RFRRequest>
```

4.185 RFRResponse

The `RFRResponse` element is an optional child of a `littleResponse` element returned in response to a `RFRRequest`.

Parent Elements:

[littleResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
response	String	Yes	The RFR Response Code indicating the result of the RFR request. minLength = N/A maxLength = 3
message	String	Yes	A brief definition of the response code returned for this transaction. minLength = N/A maxLength = 512

Child Elements:

None

4.186 routingNum

The `routingNum` element is a required child of the `echeck`, `originalAccountInfo`, and `newAccountInfo` elements defining the routing number of the Echeck account.

Type = String; **minLength** = 9; **maxLength** = 9

Parent Elements:

[echeck](#), [newAccountInfo](#), [originalAccountInfo](#), [newTokenInfo](#), [originalTokenInfo](#)

Attributes:

None

Child Elements:

None

NOTE:	If you submit an invalid routing number, Litle returns the XML Response Code 900 - Invalid Bank Routing Number.
--------------	---

4.187 RxAmount

The `RxAmount` element is an optional child of the `healthcareAmounts` element and defines the healthcare amount used for the purchased medications. The decimal is implied. Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

Optional: [healthcareAmounts](#)

Attributes:

None

Child Elements:

None

4.188 sale

The `sale` element is the parent element for all Sale transactions. A Sale transaction is a combination Authorization and Capture transaction. You can use this element in either Online or Batch transactions.

Parent Elements:

[litleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. This attribute is also used for Duplicate Transaction Detection. Failure to include a value for this attribute disables Duplicate Detection for the transaction. Please refer to Duplicate Transaction Detection on page 6 for additional information about the operation of Duplicate checking. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [amount](#), [orderSource](#), (choice of) [card](#), [paypal](#), [paypage](#), or [token](#)

NOTE: The `cardholderAuthentication` child element is required only for 3-D Secure transactions and for BML ecommerce transactions.

Optional: [litleTxnId](#), [customerInfo](#), [billToAddress](#), [shipToAddress](#), [billMeLaterRequest](#), [cardholderAuthentication](#), [customBilling](#), [taxType](#), [enhancedData](#), [processingInstructions](#), [pos](#), [paypalOrderComplete](#), [amexAggregatorData](#), [allowPartialAuth](#), [healthcareIIAS](#), [merchantData](#)

NOTE: The `fraudCheck` element has been deprecated; use the `cardholderAuthentication` element instead.

Also, the `enhancedData` element and two of its child elements, `deliveryType` and `shippingAmount`, are required for Bill Me Later Authorizations.

4.189 saleResponse

The `saleResponse` element is the parent element for information returned in response to a Sale transaction. It can be a child of either a `littleOnlineResponse` element or a `batchResponse` element.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the Sale transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the Sale transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the Sale transaction. minLength = 1 maxLength = 25
duplicate	Boolean	No	If the request is a duplicate (see Online Duplicate Checking on page 7), the response includes the duplicate flag set to true and the entire original response. Note: This attribute applies only to Online transaction responses.

Child Elements:

Required: [littleTxnId](#), [orderId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [cardProductId](#) (see Note below), [authCode](#), [authorizationResponseSubCode](#) (see Note below), [approvedAmount](#), [accountInformation](#), [fraudResult](#), [billMeLaterResponseData](#), [tokenResponse](#), [enhancedAuthResponse](#), [accountUpdater](#), [recycling](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

The `cardProductId` element returns a raw code referencing the card type. Please consult your Little Relationship Manager for additional information.

The `authorizationResponseSubCode` element is not used at this time.

4.190 salesTax

The `salesTax` element defines the amount of sales tax included in the transaction amount. Although the schema defines it as an optional child of the `enhancedData` element, it is required to receive the best interchange rate for Level II and Level III corporate purchases. The decimal is implied. Example: 500 = \$5.00.

NOTE: For a non-taxable transaction, use 0 as the value. In this case you must also set the `taxExempt` element to true.

If you provide `detailTax` data, the `salesTax` should be the sum of the `detailTax`.

Type = Integer; totalDigits = 8

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.191 sellerId

The `sellerId` element is a required child of the `amexAggregatorData` element, which defines the Seller Id as assigned by American Express.

Type = String; **minLength** = 1; **maxLength** = 16

Parent Elements:

[amexAggregatorData](#)

Attributes:

None

Child Elements:

None

4.192 sellerMerchantCategoryCode

The sellerMerchantCategoryCode element is a required child of the amexAggregatorData element, which defines the Merchant Category Code as assigned by American Express.

Type = String; **minLength** = 1; **maxLength** = 4

Parent Elements:

[amexAggregatorData](#)

Attributes:

None

Child Elements:

None

4.193 shipFromPostalCode

The `shipFromPostalCode` element defines the postal code from which the product ships in the `enhancedData` element.

Type = String; **minLength** = N/A; **maxLength** = 20

NOTE:	Although the schema specifies the maxLength of the <code><shipFromPostalCode></code> element as 20 characters, in practice you should never exceed 10 characters in your submissions.
--------------	--

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.194 shippingAmount

The `shippingAmount` element defines shipping cost for the order. Although the schema defines it as an optional child of the `enhancedData` element, it is required by Visa for Level III interchange rates. The decimal is implied. Example: 500 = \$5.00.

This element is also required for Bill Me Later transactions.

Type = Integer; **totalDigits** = 8

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.195 shipToAddress

The `shipToAddress` element contains several child elements that define the postal mailing address (and telephone number) used for shipping purposes.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [sale](#)

Attributes:

None

Child Elements: (all Optional)

[name](#), [addressLine1](#), [addressLine2](#), [addressLine3](#), [city](#), [state](#), [zip](#), [country](#), [email](#), [phone](#)

Example: `shipToAddress` Structure

```
<shipToAddress>
  <name>Customer's Full Name</name>
  <addressLine1>Address Line 1</addressLine1>
  <addressLine2>Address Line 2</addressLine2>
  <addressLine3>Address Line 3</addressLine3>
  <city>City</city>
  <state>State Abbreviation</state>
  <zip>ZIP Code</zip>
  <country>Country Code</country>
  <email>Email Address</email>
  <phone>Telephone Number</phone>
</shipToAddress>
```

4.196 ssn

The `ssn` element is an optional child of the `customerInfo` element. It is used in combination with several other elements to provide required information for some Bill Me Later transactions.

Type = Pattern; **minLength** = 4 (last four digits of SSN); **maxLength** = 9 (full SSN)

NOTE: In order for a BML transaction to succeed, you must include this element if:

- the customer does not have a BML account

or

- the customer has a BML account, but the account has not been authenticated.

You do not need to include this element if the BML account has been authenticated.

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.197 state

The `state` element defines the customer's state name in the `billToAddress`, `shipToAddress`, `taxBilling` and elements.

Type = String; **minLength** = N/A; **maxLength** = 2

NOTE:	Although the schema defines the maxLength for this element as 30, the best practice is to use the 2 character abbreviation. When submitting an eCheck Verification transaction, you must use the 2 character abbreviation or the transaction will be rejected with a 370 reason code.
--------------	--

Parent Elements:

[billToAddress](#), [shipToAddress](#), [taxExempt](#)

Attributes:

None

Child Elements:

None

4.198 taxAmount

The `taxAmount` element is a required child of the `detailTax` element and an optional child of the `lineItemData` element and defines the detail tax amount on the purchased good or service. The decimal is implied. Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

Required: [detailTax](#)

Optional: [lineItemData](#)

NOTE:	If you include <code>taxAmount</code> as a child of <code>lineItemData</code> along with <code>detailTax</code> , the <code>lineItemData taxAmount</code> should be the sum of the <code>taxAmount</code> children from <code>detailTax</code> children.
--------------	--

Attributes:

None

Child Elements:

None

4.199 taxExempt

The `taxExempt` element is an optional child of the `enhancedData` element and specifies whether or not the transaction is exempt from sales tax. If you do not include this element, the value defaults to **false**.

NOTE: You must set this element to true, if you set the `salesTax` element to 0.

Type = Boolean; **Valid values** = true or false

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.200 taxIncludedInTotal

The `taxIncludedInTotal` element is an optional child of the `detailTax` element and defines whether or not the tax is included in the total purchase amount.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[detailTax](#)

Attributes:

None

Child Elements:

None

4.201 taxRate

The `taxRate` element is an optional child of the `detailTax` element and defines the tax rate applied to this specific taxable amount.

Type = Decimal; **totalDigits** = 5

Parent Elements:

[detailTax](#)

Attributes:

None

Child Elements:

None

4.202 taxType

The `taxType` element is an optional child of several transaction types that designates the transaction as either a convenience fee or tax payment for merchants using the Visa Tax Payment Program or the MasterCard Convenience Fee Program.

Type = String (enum); **minLength** = N/A; **maxLength** = 1; **Valid Values** = payment or fee

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements:

None

4.203 taxTypeIdentifier

The `taxTypeIdentifier` element is an optional child of the `detailTax` element and defines the type of tax collected on this specific tax amount. If the tax type identifier is unknown, do not include this element.

Type = String (Enum); **minLength** = N/A; **maxLength** = 2

Parent Elements:

[detailTax](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
00	Unknown
01	Federal/National Sales Tax
02	State Sales Tax
03	City Sales Tax
04	Local Sales Tax
05	Municipal Sales Tax
06	Other Tax
10	Value Added Tax (VAT)
11	Goods and Services Tax (GST)
12	Provincial Sales Tax (PST)
13	Harmonized Sales Tax (HST)
14	Quebec Sales Tax (QST)
20	Room Tax
21	Occupancy Tax
22	Energy Tax

4.204 termsAndConditions

The `termsAndConditions` element is an optional child of the `billMeLaterRequest` element and defines the specific lending terms of the BML account.

NOTE: The Bill Me Later documentation requires you use code 12103 for Call Center purchases and code 32103 for Web purchases.
Please refer to your Bill Me Later documentation for additional information.

Type = Integer; totalDigits = 5

Parent Elements:

[billMeLaterRequest](#)

Attributes:

None

Child Elements:

None

4.205 token

The `token` element has two uses depending upon whether the element concerns a Little generated token (for tokenized merchants) or a PayPal generated token.

4.205.1 token (Little generated card number replacement)

In this case, the `token` element replaces the `card` element in tokenized card transactions or the `echeck` element in eCheck transactions, and defines the tokenized payment card/account information.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#), [accountUpdate](#)

Attributes:

None

Child Elements:

Required: [littleToken](#)

Optional: [expDate](#), [cardValidationNum](#), [routingNum](#), [accType](#)

4.205.2 token (PayPal generated)

In this case, the `token` element is the token generated by PayPal.

Type = String; **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[paypal](#)

Attributes:

None

Child Elements:

None

4.206 tokenMessage

The `tokenMessage` element provides a short, human-readable explanation of the `tokenResponseCode` (see [Table 4-2](#)).

Type = String; **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[tokenResponse](#)

Attributes:

None

Child Elements:

None

4.207 tokenResponse

The `tokenResponse` element is the parent element for several children defining the registered token, as well the either card type and BIN, or last three characters of the account number in the case of eChecks. This element appears in the response only if a tokenized merchant submits card information in the transaction request.

Parent Elements:

[authorizationResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [echeckCreditResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [forceCaptureResponse](#), [saleResponse](#)

Attributes:

None

Child Elements:

Required: [tokenResponseCode](#), [tokenMessage](#)

Optional: [littleToken](#), [type](#), [bin](#), [eCheckAccountSuffix](#)

Example: tokenResponse Structure

```
<tokenResponse>
  <littleToken>Little Token</littleToken>
  <tokenResponseCode>Response Code</tokenResponseCode>
  <tokenMessage>Response Message</tokenMessage>
  <type>Method of Payment</type>
  <bin>BIN</bin>
  <eCheckAccountSuffix>Last 3 of Account Number</eCheckAccountSuffix> (returned for eCheck account tokens)
</tokenResponse>
```

4.208 tokenResponseCode

The `tokenResponseCode` element provides a 3-digit code (see [Table 4-2](#)) indicating the results of a transaction involving the conversion or attempted conversion of an account number to a token. The `tokenMessage` element contains a short, human-readable explanation of the `tokenResponseCode`.

Type = String; **minLength** = N/A; **maxLength** = 3

Parent Elements:

[tokenResponse](#)

Attributes:

None

Child Elements:

None

TABLE 4-2 tokenResponseCode and tokenMessage Values

Code	Message
801	Account number was successfully registered
802	Account number was previously registered
820	Credit card number was invalid
821	Merchant is not authorized for tokens
822	Token was not found
823	Token was Invalid
898	Generic token registration error
899	Generic token use error

4.209 totalHealthcareAmount

The `totalHealthcateAmount` element is a required child of the `healthcareAmounts` element and defines the total amount of healthcare related purchases. This value must be the sum of the values applied to the following elements: `RxAmount`, `visionAmount`, `clinicOtherAmount`, and `dentalAmount`. The decimal is implied. Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

Optional: [healthcareAmounts](#)

Attributes:

None

Child Elements:

None

4.210 track

The `track` element is child of the `card` element, which is required for card-present transactions. The contents of the `track` element is the data read from the magnetic stripe.

Type = String; **minLength** = 1; **maxLength** = 256

Parent Elements:

[card](#)

Attributes:

None

Child Elements:

None

4.211 transactionId

The `transactionId` element is a required child of the `paypal` element, specifying the transaction Id returned from PayPal.

NOTE: The value of the Litle & Co. `<transactionId>` element must match the TRANSACTIONID returned by the `DoExpressCheckoutPayment` call operation to PayPal.

Type = String; minLength = N/A; maxLength = N/A

Parent Elements:

[paypal](#)

Attributes:

None

Child Elements:

None

4.212 type

The `type` element has two uses in LittleXML depending upon the parent. In one case it defines the type of account used in the transaction in terms of association, company, Bill Me Later, PayPal, or eCheck. When used as a child of the `fundingSource` element, it defines the card type in terms of prepaid, credit, debit, FSA, or unknown.

4.212.1 type Element as a child of the parent elements listed below

This `type` element defines the type of account used in the transaction in terms of card association, card company, Bill Me Later, PayPal, or eCheck.

Type = String (Enum); **minLength** = N/A; **maxLength** = 2

Parent Elements:

[accountInformation](#), [newCardInfo](#), [newCardTokenInfo](#), [originalCard](#), [originalCardInfo](#), [originalCardTokenInfo](#), [originalToken](#), [updatedCard](#), [updatedToken](#), [registerTokenResponse](#), [tokenResponse](#), [card](#), [paypage](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
MC	MasterCard
VI	Visa
AX	American Express
DC	Diner's Club
DI	Discover
PP	PayPal
JC	JCB (Japanese Credit Bureau)
BL	Bill Me Later
EC	eCheck

Enumeration	Description
"" (empty)	Card type unknown or undefined

4.212.2 type Element as a child of fundingSource

This `type` element defines the card type in terms of prepaid, credit, debit, FSA, or unknown.

Type = String (Enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[fundingSource](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
UNKNOWN	The card type can not be determined.
PREPAID	This is a prepaid card.
CREDIT	This is a credit card.
DEBIT	This is a debit card.
FSA	This is a Flexible Spending Account card. Cards of this type can be used only for IRS-approved healthcare items.

NOTE: The `fundingSource` element and its child elements, `type` and `availableBalance` are associated with the Smart Authorization features (see [Smart Authorization Features](#) on page 11.)

Please consult your Relationship Manager for additional information.

4.213 unitCost

The `unitCost` element is an optional child of the `lineItemData` element, which specifies the price of one unit of the item purchased. Although the schema defines it as an optional child of the `enhancedData` element, it is required by Visa for Level III interchange rates. The value must be greater than or equal to 0.

Type = Decimal; **minInclusive value** = 0, **totalDigits** = 12

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.214 unitOfMeasure

The `unitOfMeasure` element is an optional child of the `lineItemData` element, which specifies the unit of measure of the purchased item. For example, each, kit, pair, gallon, and month would all be valid values. Although an optional element, it is required by Visa and MasterCard when specifying line item data.

Type = String; **minLength** = 1; **maxLength** = 12

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.215 updatedCard

The `updatedCard` element is an optional child of the `accountUpdateResponse` element, which contains child elements providing the updated information for the submitted card.

Parent Elements:

[accountUpdateResponse](#)

Attributes:

None

Child Elements:

[type](#), [number](#), [expDate](#)

Example: updatedCard Structure

```
<updatedCard>
  <type>Card Type</type>
  <number>New Account Number</number>
  <expDate>New Expiration Date</expDate>
</updatedCard>
```

4.216 updatedToken

The `updatedToken` element is an optional child of the `accountUpdateResponse` element, which contains child elements providing the updated information for the submitted token.

Parent Elements:

[accountUpdateResponse](#)

Attributes:

None

Child Elements:

[type](#), [number](#), [expDate](#), [bin](#)

Example: originalCard Structure

```
<updatedToken>
  <littleToken>New Token Number</littleToken>
  <expDate>New Expiration Date</expDate>
  <type>Card Type</type>
  <bin>Card BIN</bin>
</updatedToken>
```


4.217 url

The `url` element is an optional child of the `customBilling` element. You use it to designate your customer service web site instead of providing a customer service phone number. This element may include any of the following characters: A-Z, a-z, 0-9, /, \, -, ., or _.

Type = String; **minLength** = N/A; **maxLength** = 13

NOTE: Please consult your Little Relationship Manager prior to attempting to use the `<url>` element. This contents of this element are discarded unless you are specifically enabled to use in your LittleXML submissions.

Parent Elements:

[customBilling](#)

Attributes:

None

Child Elements:

None

4.218 user

The `user` element is a required child of the `authentication` element. It is a unique identifier of the user/merchant used to authenticate that the message is from a valid source.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[authentication](#)

Attributes:

None

Child Elements:

None

4.219 verificationCode

NOTE: This element is not used at this time.

The `verificationCode` element is an optional child of the `echeckSaleResponse` element. It specifies the verification code from the associated eCheck Sale transaction.

Type = String; **minLength** = N/A; **maxLength** = 6

Parent Elements:

[echeckSalesResponse](#)

Attributes:

None

Child Elements:

None

4.220 verify

The `verify` element is an optional child of the `echeckSale` element, which allows you to specify to perform an eCheck Verification prior to processing the sale. If the account fails the verification operation, the system does not process the sale.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[echeckSale](#)

Attributes:

None

Child Elements:

None

4.221 visionAmount

The `visionAmount` element is an optional child of the `healthcareAmounts` element and defines the healthcare amount used for vision related purchases. The decimal is implied.

Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 8

Parent Elements:

Optional: [healthcareAmounts](#)

Attributes:

None

Child Elements:

None

4.222 virtualAuthenticationKeyData

The `virtualAuthenticationKeyData` is an optional child of the `billMeLaterRequest` element.

Type = String; **minLength** = N/A; **maxLength** = 4

Parent Elements:

[billMeLaterRequest](#)

Attributes:

None

Child Elements:

None

4.223 virtualAuthenticationKeyPresenceIndicator

The `virtualAuthenticationKeyPresenceIndicator` is an optional child of the `billMeLaterRequest` element.

Type = String; **minLength** = N/A; **maxLength** = 1

Parent Elements:

[batchRequest](#)

Attributes:

None

Child Elements:

None

4.224 void

The `void` element is the parent element for all Void transactions. You can use this element only in Online transactions. If you use this Litle Recycling Engine, you can use the `void` transaction to halt the recycling of a `sale` transaction.

Parent Elements:

[litleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. This attribute is also used for Duplicate Transaction Detection. Failure to include a value for this attribute disables Duplicate Detection for the transaction. Please refer to Duplicate Transaction Detection on page 6 for additional information about the operation of Duplicate checking. minLength = N/A maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	Required attribute that defines the merchant sub-group in the user interface where this transaction will be displayed. Please refer to Coding for Report Groups on page 9 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [litleTxnId](#)

Optional: [processingInstructions](#)

4.225 voidResponse

The `voidResponse` element is the parent element for information returned to you in response to a Void transaction.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	The response returns the same value submitted in the void transaction. minLength = N/A maxLength = 25
customerId	String	No	The response returns the same value submitted in the void transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the void transaction. minLength = 1 maxLength = 25
duplicate	Boolean	No	If the request is a duplicate (see Online Duplicate Checking on page 7), the response includes the duplicate flag set to true and the entire original response. Note: This attribute applies only to Online transaction responses.

Child Elements: (all Required)

[littleTxnId](#), [response](#), [responseTime](#), [message](#), [postDate](#)

4.226 yearsAtEmployer

The `yearsAtEmployer` element is an optional child of the `customerInfo` element and defines the number of years the customer has worked for their current employer. It is used in combination with several other elements to provide required information for some Bill Me Later transactions.

Type = Integer; **totalDigits** = 2

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.227 yearsAtResidence

The `yearsAtResidence` element is an optional child of the `customerInfo` element and defines the number of years the customer has resided in their current domicile. It is used in combination with several other elements to provide required information for some Bill Me Later transactions.

Type = Integer; **totalDigits** = 2

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.228 zip

The `zip` element defines the customer's postal code in both the `billToAddress` and `shipToAddress` elements.

Type = String; **minLength** = N/A; **maxLength** = 20

NOTE:	Although the schema specifies the maxLength of the <code><zip></code> element as 20 characters, in practice you should never exceed 10 characters in your submissions.
--------------	---

Parent Elements:

[billToAddress](#), [shipToAddress](#)

Attributes:

None

Child Elements:

None



PAYMENT TRANSACTION RESPONSE CODES

This appendix provides reference material regarding the codes that are returned in a LittleXML response for a payment transaction. This appendix contains the following sections:

- [Payment Transaction Response Codes](#)
- [Visa Authentication Result Codes](#)
- [AVS Response Codes](#)
- [AAVS Response Codes](#)
- [Card Validation Response Codes](#)
- [XML Validation Error Messages](#)
- [eCheck Return Reason Codes](#)

A.1 Payment Transaction Response Codes

This section contains a list of codes and messages that the system can return in the response message for a payment transaction.

NOTE: For information concerning Chargeback Response Code, see the *Chargeback XML and Support Documentation API Reference Guide*.

Table A-1 shows all possible values for the <response> and <message> elements. You should code appropriately to handle all codes applicable to the transactions you use.

- The Response Code value appears in the <response> element.
- The Response Message value appears in the <message> element.

TABLE A-1 Valid Values for the Response and Message Elements

Response Code	Response Message	Response Type	Description
000	Approved	Approved	No action required.
010	Partially Approved	Approved	The authorized amount is less than the requested amount.
100	Processing Network Unavailable	Soft Decline	There is a problem with the card network. Contact the network for more information.
101	Issuer Unavailable	Soft Decline	There is a problem with the issuer network. Please contact the issuing bank.
102	Re-submit Transaction	Soft Decline	There is a temporary problem with your submission. Please re-submit the transaction.
110	Insufficient Funds	Soft Decline	The card does not have enough funds to cover the transaction.
111	Authorization amount has already been depleted	Hard Decline	The total amount of the original Authorization has been used.
120	Call Issuer	Referral or Soft Decline	There is an unspecified problem, contact the issuing bank.
121	Call AMEX	Referral	There is an unspecified problem; contact AMEX.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
122	Call Diners Club	Referral	There is an unspecified problem; contact Diners Club.
123	Call Discover	Referral	There is an unspecified problem; contact Discover.
124	Call JBS	Referral	There is an unspecified problem; contact JBS.
125	Call Visa/MasterCard	Referral	There is an unspecified problem; contact Visa or MasterCard.
126	Call Issuer - Update Cardholder Data	Referral	Some data is out of date; contact the issuer to update this information.
127	Exceeds Approval Amount Limit	Hard Decline	This transaction exceeds the daily approval limit for the card.
130	Call Indicated Number	Referral	There is an unspecified problem; contact the phone number provided.
140	Update Cardholder Data	Referral	Cardholder data is incorrect; contact the issuing bank.
191	The merchant is not registered in the update program.	N/A	This is an Account Updater response indicating a set-up problem that must be resolved prior to submitting another request file. Escalate this to your Little Relationship Manager.
192	Merchant not certified/enabled for IIAS	Hard Decline	Your organization is not certified or enabled for IIAS/FSA transactions.
301	Invalid Account Number	Hard Decline	The account number is not valid; contact the cardholder to confirm information or inquire about another form of payment.
302	Account Number Does Not Match Payment Type	Hard Decline	The payment type was selected as one card type (e.g. Visa), but the card number indicates a different card type (e.g. MasterCard).
303	Pick Up Card	Hard Decline	This is a card present response, but in a card not present environment. Do not process the transaction and contact the issuing bank.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
304	Lost/Stolen Card	Hard Decline	The card has been designated as lost or stolen; contact the issuing bank.
305	Expired Card	Hard Decline	The card is expired.
306	Authorization has expired; no need to reverse	Hard Decline	The original Authorization is no longer valid, because it has expired. You can not perform an Authorization Reversal for an expired Authorization.
307	Restricted Card	Hard Decline	The card has a restriction preventing approval for this transaction. Please contact the issuing bank for a specific reason.
308	Restricted Card - Chargeback	Hard Decline	The card has a restriction preventing approval if there are any chargebacks against it.
309	Restricted Card - Prepaid Card Filtering Service	Hard Decline	This transaction is being declined due the operation of the Little Prepaid Card Filtering service.
310	Invalid track data	Hard Decline	The track data is not valid.
311	Deposit is already referenced by a chargeback	Hard Decline	The deposit is already referenced by a chargeback; therefore, a refund cannot be processed against the original transaction.
312	Restricted Card - International Card Filtering Service	Hard Decline	This transaction is being declined due the operation of the Little International Card Filtering Service.
315	Restricted Card - Auth Fraud Velocity Filtering Service	Hard Decline	This transaction is being declined due the operation of the Little Auth Fraud Velocity Filtering Service.
316	Automatic Refund Already Issued	Hard Decline	This refund transaction is a duplicate for one already processed automatically by the Little Fraud Chargeback Prevention Service (FCPS).

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
320	Invalid Expiration Date	Hard Decline	The expiration date is invalid
321	Invalid Merchant	Hard Decline	The card is not allowed to make purchases from this merchant (e.g. a Travel only card trying to purchase electronics).
322	Invalid Transaction Note: If you are enabled for Transaction Filtering, but have not upgraded to use schema version 8.3 or above, the system returns this code for transactions filtered by the Prepaid or International Card Filtering Service. Also, if you are enabled for Velocity Fraud Filtering, but have not upgraded to V8.9, you will receive this code for filtered transactions.	Hard Decline	The transaction is not permitted; contact the issuing bank.
323	No such issuer	Hard Decline	The card number references an issuer that does not exist. Do not process the transaction.
324	Invalid Pin	Hard Decline	The PIN provided is invalid.
325	Transaction not allowed at terminal	Hard Decline	The transaction is not permitted; contact the issuing bank.
326	Exceeds number of PIN entries	Hard Decline	(Referring to a debit card) The incorrect PIN has been entered excessively and the card is locked.
327	Cardholder transaction not permitted	Hard Decline	Merchant does not allow that card type or specific transaction.
328	Cardholder requested that recurring or installment payment be stopped	Hard Decline	Recurring/Installment Payments no longer accepted by the card issuing bank.
330	Invalid Payment Type	Hard Decline	This payment type is not accepted by the issuer.
335	This method of payment does not support authorization reversals	Hard Decline	You can not perform an Authorization Reversal transaction for this payment type.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
336	Reversal amount does not match Authorization amount.	Hard Decline	For a merchant initiated reversal against an American Express authorization, the reversal amount must match the authorization amount exactly.
340	Invalid Amount	Hard Decline	The transaction amount is invalid (too high or too low). For example, less than 0 for an authorization, or less than .01 for other payment types.
341	Invalid Healthcare Amounts	Hard Decline	The amount submitted with this FSA/Healthcare transaction is invalid. The FSA amount must be greater than 0, and cannot be greater than the transaction amount.
346	Invalid billing descriptor prefix	Hard Decline	The billing descriptor prefix submitted is not valid.
347	Invalid billing descriptor	Hard Decline	The billing descriptor is not valid because you are not authorized to send transactions with custom billing fields.
349	Do Not Honor	Soft Decline	The issuing bank has put a temporary hold on the card.
350	Generic Decline	Soft or Hard Decline	There is an unspecified problem; contact the issuing bank for more details. Note: This code can be a hard or soft decline, depending on the method of payment, and other variables.
351	Decline - Request Positive ID	Hard Decline	Card Present transaction that requires a picture ID match.
352	Decline CVV2/CID Fail	Hard Decline	The CVV2/CID is invalid.
354	3-D Secure transaction not supported by merchant	Hard Decline	You are not certified to submit 3-D Secure transactions.
356	Invalid purchase level III, the transaction contained bad or missing data	Soft Decline	Submitted Level III data is bad or missing.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
357	Missing healthcareIIAS tag for an FSA transaction	Hard Decline	The FSA Transactions submitted does not contain the <code><healthcareIIAS></code> data element.
360	No transaction found with specified litleTxnId	Hard Decline	There were no transactions found with the specified litleTxnId.
361	Authorization no longer available	Hard Decline	The authorization for this transaction is no longer available; the authorization has already been consumed by another capture.
362	Transaction Not Voided - Already Settled	Hard Decline	This transaction cannot be voided; it has already been delivered.
363	Auto-void on refund	Hard Decline	This transaction (both capture and refund) has been voided.
364	Invalid Account Number - original or NOC updated eCheck account required	Hard Decline	The submitted account number is invalid. Confirm the original account number or check NOC for new account number.
365	Total credit amount exceeds capture amount	Hard Decline	The amount of the credit is greater than the capture, or the amount of this credit plus other credits already referencing this capture are greater than the capture amount.
366	Exceed the threshold for sending redeposits	Hard Decline	NACHA rules allow two redeposit attempts within 180 days of the settlement date of the initial deposit attempt. This threshold has been exceeded.
367	Deposit has not been returned for insufficient/non-sufficient funds	Hard Decline	NACHA rules only allow redeposit attempts against deposits returned for Insufficient or Uncollected Funds.
368	Invalid check number	Soft Decling	The check number is invalid.
369	Redeposit against invalid transaction type	Hard Decline	The redeposit attempted against an invalid transaction type.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
370	Internal System Error - Call Little	Hard Decline	There is a problem with the Little System. Contact support@little.com.
372	Soft Decline - Auto Recycling In Progress	Soft Decline	The transaction was intercepted because it is being auto recycled by the Recycling Engine.
373	Hard Decline - Auto Recycling Complete	Hard Decline	The transaction was intercepted because auto recycling has completed with a final decline.
500	The account number was changed	Hard Decline	An Account Updater response indicating the Account Number changed from the original number.
501	The account was closed	Hard Decline	An Account Updater response indicating the account was closed. Contact the cardholder directly for updated information.
502	The expiration date was changed	N/A	An Account Updater response indicating the Expiration date for the card has changed.
503	The issuing bank does not participate in the update program	N/A	An Account Updater response indicating the issuing bank does not participate in the update program
504	Contact the cardholder for updated information	N/A	An Account Updater response indicating you should contact the cardholder directly for updated information.
505	No match found	N/A	An Account Updater response indicating no match was found in the updated information.
506	No changes found	N/A	An Account Updater response indicating there have been no changes to the account information.
601	Soft Decline - Primary Funding Source Failed	Soft Decline	A PayPal response indicating the transaction failed due to an issue with primary funding source (e.g. expired Card, insufficient funds, etc.).

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
602	Soft Decline - Buyer has alternate funding source	Soft Decline	A PayPal response indicating the merchant may resubmit the transaction immediately, and the use of an alternate funding source will be attempted.
610	Hard Decline - Invalid Billing Agreement Id	Hard Decline	A PayPal response indicating the Billing Agreement ID is invalid.
611	Hard Decline - Primary Funding Source Failed	Hard Decline	A PayPal response indicating the issuer is unavailable.
612	Hard Decline - Issue with Paypal Account	Hard Decline	A PayPal response indicating the transaction failed due to an issue with the buyer account.
613	Hard Decline - PayPal authorization ID missing	Hard Decline	A PayPal response indicating the need to correct the authorization ID before resubmitting.
614	Hard Decline - confirmed email address is not available	Hard Decline	A PayPal response indicating your account is configured to decline transactions without a confirmed address. request another payment method or contact support@litle.com to modify your account settings.
615	Hard Decline - PayPal buyer account denied	Hard Decline	A PayPal response indicating account unauthorized payment risk.
616	Hard Decline - PayPal buyer account restricted	Hard Decline	A PayPal response indicating PayPal is unable to process the payment. Buyer should contact PayPal with questions.
617	Hard Decline - PayPal order has been voided, expired, or completed	Hard Decline	A PayPal response indicating no further authorizations/captures can be processed against this order. A new order must be created.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
618	Hard Decline - issue with PayPal refund	Hard Decline	A PayPal response indicating one of these potential refund related issues: duplicate partial refund must be less than or equal to original or remaining amount, past time limit, not allowed for transaction type, consumer account locked/inactive, or complaint exists - only a full refund of total/remaining amount allowed. Contact support@litle.com for specific details.
619	Hard Decline - PayPal credentials issue	Hard Decline	A PayPal response indicating you do not have permissions to make this API call.
620	Hard Decline - PayPal authorization voided or expired	Hard Decline	A PayPal response indicating you cannot capture against this authorization. You need to perform a brand new authorization for the transaction.
621	Hard Decline - required PayPal parameter missing	Hard Decline	A PayPal response indicating missing parameters are required. Contact support@litle.com for specific details.
622	Hard Decline - PayPal transaction ID or auth ID is invalid	Hard Decline	A PayPal response indicating the need to check the validity of the authorization ID prior to reattempting the transaction.
623	Hard Decline - Exceeded maximum number of PayPal authorization attempts	Hard Decline	A PayPal response indicating you should capture against a previous authorization.
624	Hard Decline - Transaction amount exceeds merchant's PayPal account limit.	Hard Decline	A PayPal response indicating the transaction amount exceeds the merchant's account limit. Contact support@litle.com to modify your account settings.
625	Hard Decline - PayPal funding sources unavailable.	Hard Decline	A PayPal response indicating the buyer needs to add another funding sources to their account.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
626	Hard Decline - issue with PayPal primary funding source.	Hard Decline	A PayPal response indicating there are issues with the buyer's primary funding source.
627	Hard Decline - PayPal profile does not allow this transaction type.	Hard Decline	Contact Little to adjust your PayPal merchant profile preferences.
628	Internal System Error with PayPal - Contact Little	Hard Decline	There is a problem with Little's username and password. Contact support@litle.com.
629	Hard Decline - Contact PayPal consumer for another payment method	Hard Decline	A PayPal response indicating you should contact the consumer for another payment method.
701	Under 18 years old	Hard Decline	A Bill Me Later (BML) response indicating the customer is under 18 years of age based upon the date of birth.
702	Bill to outside USA	Hard Decline	A BML response indicating the billing address is outside the United States.
703	Bill to address is not equal to ship to address	Hard Decline	A BML response indicating that the billing address does not match the shipping address.
704	Declined, foreign currency, must be USD	Hard Decline	A BML response indicating the transaction is declined, because it is not in US dollars.
705	On negative file	Hard Decline	A BML response indicating the account is on the negative file.
706	Blocked agreement	Hard Decline	A BML response indicating a blocked agreement account status.
707	Insufficient buying power	Other	A BML response indicating that the account holder does not have sufficient credit available for the transaction amount.
708	Invalid Data	Hard Decline	A BML response indicating that there are one or more problems with the submitted data.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
709	Invalid Data - data elements missing	Hard Decline	A BML response indicating one or more required data elements are missing.
710	Invalid Data - data format error	Hard Decline	A BML response indicating that some data was formatted incorrectly.
711	Invalid Data - Invalid T&C version	Hard Decline	A BML response indicating the T&C version is invalid.
712	Duplicate transaction	Hard Decline-	A BML response indicating that the transaction is a duplicate.
713	Verify billing address	Hard Decline	A BML response indicating that you should verify the billing address.
714	Inactive Account	Hard Decline	A BML response indicating the customer account is inactive.
716	Invalid Auth	Hard Decline	A BML response indicating that the referenced authorization is invalid.
717	Authorization already exists for the order	Hard Decline	A BML response indicating that an authorization already exists for the transaction.
801	Account number was successfully registered	Approved	The card number was successfully registered and a token number was returned.
802	Account number was previously registered	Approved	The card number was previously registered for tokenization.
820	Credit card number was invalid	Hard Decline	The card number submitted for tokenization is invalid.
821	Merchant is not authorized for tokens	Hard Decline	Your organization is not authorized to use tokens.
822	Token was not found	Hard Decline	The token number submitted with this transaction was not found.
850	Tax Billing only allowed for MCC 9311	Hard Decline	Tax Billing elements are allowed only for MCC 9311.
851	Incomplete Tax Billing	Hard Decline	Missing taxType element

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
877	Invalid Pay Page Registration Id	Hard Decline	A Pay Page response indicating that the Pay Page Registration ID submitted is invalid.
878	Expired Pay Page Registration Id	Hard Decline	A Pay Page response indicating that the Pay Page Registration ID has expired (Pay Page Registration IDs expire 24 hours after being issued).
879	Merchant is not authorized for Pay Page	Hard Decline	Your organization is not authorized to use the Pay Page.
898	Generic token registration error	Soft Decline	There is an unspecified token registration error; contact Little & Co.
899	Generic token use error	Soft Decline	There is an unspecified token use error; contact Little & Co.
900	Invalid Bank Routing Number	Hard Decline	The eCheck routing number submitted with this transaction has failed validation.
950	Decline - Negative Information on File	Hard Decline	An eCheck response indicating the account is on the negative file.
951	Absolute Decline	Hard Decline	An eCheck response indicating that this transaction was declined.
952	The Merchant Profile does not allow the requested operation	Hard Decline	An eCheck response indicating that your Merchant Profile does not allow the requested operation. Contact your Little & Co. Relationship Manager for additional information.
953	The account cannot accept ACH transactions	Hard Decline	An eCheck response indicating the customer's checking account does not accept ACH transactions.
954	The account cannot accept ACH transactions or site drafts	Hard Decline	An eCheck response indicating the customer's checking account does not accept ACH transactions or site drafts.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
955	Amount greater than limit specified in the Merchant Profile	Hard Decline	An eCheck response indicating that the dollar amount of this transaction exceeds the maximum amount specified in your Merchant Profile. Contact your Litle & Co. Relationship Manager for additional information.
956	Merchant is not authorized to perform eCheck Verification transactions	Hard Decline	An eCheck response indicating that your organization is not authorized to perform eCheck verifications. Contact your Litle & Co. Relationship Manager for additional information.
957	First Name and Last Name required for eCheck Verifications	Hard Decline	An eCheck response indicating that the first and last name of the customer is required for eCheck verifications.
958	Company Name required for corporate account for eCheck Verifications	Hard Decline	An eCheck response indicating that the company name is required for verifications on corporate accounts.

A.2 Visa Authentication Result Codes

Table A-2 contains a list of valid authentication result codes returned by Visa. It specifies what authentication result values apply to what order sources.

TABLE A-2 Authentication Result Codes

Authentication Result Code	Description
Order Source - Ecommerce	
Blank	Standard ecommerce or non-ecommerce transactions, not an authentication or attempted authentication. CAVV not present
Order Source - any	
B	CAVV passed verification, but no liability shift because a) ECI was not 5 or 6 or b) the card type is an excluded (e.g., Commercial Card)
Order Source - 3DSAuthenticated or 3DSAttempted	
0	CAVV data field not properly formatted; verification cannot be performed.
6	CAVV not verified because Issuer has requested no verification. VisaNet processes as if CAVV is valid.
Order Source - 3DSAuthenticated	
1	CAVV failed verification
2	CAVV passed verification
D	Issuer elected to return CAVV verification results and Field 44.13 blank. Value is set by VisaNet; means CAVV Results are valid.
Order Source - 3DSAttempted	
3	CAVV passed verification
4	CAVV failed verification
5	Not currently used
7	CAVV failed verification
8	CAVV passed verification
9	CAVV failed verification; Visa generated CAVV because Issuer ACS was not available
A	CAVV passed verification; Visa generated CAVV because Issuer ACS was not available

TABLE A-2 Authentication Result Codes (Continued)

Authentication Result Code	Description
B	CAVV passed verification but no liability shift because a) ECI was not 5 or 6 or b) the card type is an excluded (e.g., Commercial Card)
C	Issuer elected to return CAVV verification results and Field 44.13 blank. Value is set by VisaNet; means CAVV Results are valid

A.3 AVS Response Codes

Table A-3 contains a list of AVS response codes that can be returned in the response for a payment transaction. There are some codes that you may never receive. Code your system to expect codes from this list. The description is not included in the response.

TABLE A-3 AVS Response Codes

AVS Response Code	Description
00	5-Digit zip and address match
01	9-Digit zip and address match
02	Postal code and address match
10	5-Digit zip matches, address does not match
11	9-Digit zip matches, address does not match
12	Zip does not match, address matches
13	Postal code does not match, address matches
14	Postal code matches, address not verified
20	Neither zip nor address match
30	AVS service not supported by issuer
31	AVS system not available
32	Address unavailable
33	General error
34	AVS not performed
40	Address failed Little & Co. edit checks

A.4 AAVS Response Codes

Table A-4 contains a list of American Express Advanced AVS response codes that can be returned as verification of information supplied in the <name>, <phone> and/or <email> child elements of the <billToAddress> element. The system returns the AAVS response code in the <advancedAVSResult> child of the <fraudResult> element.

The code returned has the following format:

- **1st position** - name match
- **2nd position** - phone match
- **3rd position** - email match
- Each position can have one of the following values:
 - **0** - No Match (failure)
 - **1** - Match
 - **2** - Not Sent
 - **3** - No Response (unchecked, retry, or service not allowed)

For example, a code of 210 would indicate that the name was not sent, the phone matches, and the email does not match.

You should code your system to parse all codes from this list. The description is not included in the response.

TABLE A-4 Advances AVS Response Codes

AAVS Response Code	Description
000	No Match
001	Email matches, name and phone do not match
002	Name and phone do not match, email not sent
003	Name and phone do not match, no response for email
010	Phone matches, name and email do not match
011	Phone and email match, name does not match
012	Phone matches, name does not match, email not sent
013	Phone matches, name does not match, no response for email
020	Name and email do not match, phone not sent

TABLE A-4 Advances AVS Response Codes

AAVS Response Code	Description
021	Email matches, name does not match, phone not sent
030	Name and email do not match, no response for phone
031	Email matches, name does not match, no response for phone
033	Name does not match, no response for phone or email
100	Name matches, phone and email do not match
101	Name and email match, phone does not match
102	Name matches, phone does not match, email not sent
103	Name matches, phone does not match, no response for email
110	Name and phone match, no match for email
111	Full match
112	Name and phone match, email not sent
113	Name and phone match, no response for email
120	Name matches, email does not match, phone not sent
121	Name and email match, phone not sent
130	Name matches, email does not match, no response for phone
131	Name and email match, no response for phone
133	Name matches, no response for phone or email
200	Name not sent, phone and email do not match
201	Email matches, phone does not match, name not sent
202	Phone does not match, name and email not sent
203	Phone does not match, name not sent, no response for email
210	Phone matches, email does not match, name not sent
211	Phone and email match, name not sent
212	Phone matches, name and email not sent
213	Phone matches, name not sent, no response for email
220	Email does not match, name and phone not sent

TABLE A-4 Advances AVS Response Codes

AAVS Response Code	Description
221	Email matches, name and phone not sent
230	Email does not match, name not sent, no response for phone
231	Email matches, name not sent, no response for phone
233	Name not sent, no response for phone and email
300	Phone and email do not match, no response for name
301	Email matches, phone does not match, no response for name
302	Phone does not match, no response for name, email not sent
303	Phone does not match, no response for name and email
310	Phone matches, email does not match, no response for name
311	Phone and email match, no response for name
312	Phone matches, email not sent, no response for name
313	Phone matches, no response for name and email
320	Email does not match, phone not sent, no response for name
321	Email matches, phone not sent, no response for name
330	Email does not match, no response for name and phone
331	Email matches, no response for name and phone
333	No response

A.5 Card Validation Response Codes

Table A-5 contains a normalized list of response codes that can be returned when requesting a card validation check.

- CVV2
- CVC2
- CID

The description is not included in the response.

NOTE: You must register your organization with American Express to use the CID service.

TABLE A-5 Card Validation Response Codes

CVV2/CVC2/CID Response Code	Description
M	Match
N	No Match
P	Not Processed
S	CVV2/CVC2/CID should be on the card, but the merchant has indicated CVV2/CVC2/CID is not present
U	Issuer is not certified for CVV2/CVC2/CID processing
"" (empty response)	Check was not done for an unspecified reason

A.6 XML Validation Error Messages

Table A-6 provides examples of XML Validation Error Messages. These messages are the value associated with the `message` attribute of either a `littleResponse` or `littleOnlineResponse`, when the `response="1"` (the `response` attribute).

NOTE: If `response="0"`, the associated `message="Valid Format"`.

TABLE A-6 Response Header Error Message Examples

Example Message (message attribute of <code>littleOnlineResponse</code>)	Description (line numbers will vary according to the location of the error)
Error validating xml data against the schema on line 13. The length of the value is 3, but the required minimum is 4.	The value on line 13 does not meet the minimum length requirement for the element as specified in the schema. For example, if you specified 812 as a value for the <code><expDate></code> element, which has a <code>minLength</code> of 4, the system returns this error message.
Error validating xml data against the schema on line 18. The length of the value is 6, but the required maximum is 4.	The value on line 18 exceeds the maximum length requirement for the element as specified in the schema. For example, if you specified 082012 as a value for the <code><expDate></code> element, which has a <code>maxLength</code> of 4, the system returns this error message.
Error validating xml data against the schema on line 11. The value is not a member of the enumeration.	The value on line 11 is not a valid enumeration for the specified element. For example, The <code><type></code> element allows values of VI, MC, DI, AX, DC, JC, PP and BL. If you submitted a value of VISA, the system returns this error message.
Error validating xml data against the schema on line 8. Content of element <code><amount></code> is incomplete.	The <code><amount></code> element does not contains a valid value. For example, if you submitted a <code>captureGivenAuth</code> request and included the <code>amount</code> element without specifying a value, the system returns this error message.
Error validating xml data against the schema on line 6 tag name <code><echeckSale></code> is not allowed. Possible tag names are: <authorization>,<capture>,<credit>,<sale>,<void>;	The submitted transaction failed validation against the schema, because an element name was out of sequence or not allowed in the transaction. The error message specifies the invalid element (<code><echeckSale></code> in the example), as well as the possible valid elements.

TABLE A-6 Response Header Error Message Examples (Continued)

Example Message (message attribute of <code>littleOnlineResponse</code>)	Description (line numbers will vary according to the location of the error)
System Error - Call Little & Co.	<p>Typically, the system returns this error if there was a problem with authentication due to an error in the submitted Merchant Id, user, and/or password.</p> <p>The problem may also be due to the use of single quotes around the attribute (<code>merchantId</code>) value.</p>
Error validating xml data against the schema on line 1. Probably namespace URI of tag <code><littleOnlineRequest></code> is wrong (correct one is <code><http://www.little.com/schema></code> ;))	<p>The URI named in the <code>xmlns=</code> attribute is incorrect. The problem may also be due to the use of single quotes around the attribute value.</p> <p>Note: The URI may differ based upon the version of Little XML you are using.</p>
Error validating xml data against the schema on line 12786. The entity name must immediately follow the <code>& amp;</code> in the entity reference.	<p>The <code>'&'</code> symbol is used in XML to designate certain special characters. The error indicates that the symbol was submitted without an entity name (for example, <code>&quot;</code>; or <code>& amp;</code>;)).</p> <p>Typically, the error occurs when the name or one of the address lines of the <code>billToAddress</code> element includes the symbol instead of the entity reference. For example, "John & Mary Smith" should be sent as "John & amp; Mary Smith" or "John and Mary Smith").</p>
Error validating xml data against the schema on line 1. Content is not allowed in prolog.	<p>This error is usually an indication of extraneous characters appearing in front of the first XML element. For example, the <code>"?"</code> before the <code>"<"</code> symbol in the following line would cause this error to be returned:</p> <pre>?<?xml version="1.0" encoding="utf-8"></pre>
Duplicate Batch (Little ID: 28292109643, session sequence: 1, unique ID:) not processed - 29 duplicate transactions (57 total) in a row found. Duplicate Batch (Little ID: 23829210964, session sequence: 1, unique ID:) not processed - 96.49% of the transactions (57 total) in the batch are duplicates.	<p>(Batch file only) The system has determined that the Batch file is a duplicate and therefore not processed.</p> <p>The first part of the message provides the count of the greatest number of consecutive duplicate transaction in the batch (29 in the example). The second part of the message the overall percentage of duplicates in the batch (96.49% in the example).</p> <p>The limits are more than 10 consecutive duplicate transactions detected and/or more than 25% of all transaction in the batch detected as duplicates.</p>

A.7 eCheck Return Reason Codes

Table A-7 is a list of eCheck Return Reason Codes. These codes are not returned in the LitleXML response messages. The codes are visible in the Litle Merchant Accounting System UI on the eCheck Returns Received report, as well as the Payment Detail screen.

NOTE: If an eCheck is returned for reason Code R01 or R09, it is eligible for redeposit.

TABLE A-7 eCheck Return Reason Codes

eCheck Return Reason Code	Description
R01	Insufficient funds in account
R02	Account is closed
R03	No account on file
R04	Invalid account number
R05	Unauthorized debit to consumer account
R06	Returned at request of ODFI
R07	Authorization revoked by customer
R08	Payment stopped
R09	Insufficient collected funds in account being charged
R10	Customer advises not Authorized, notice not provided, improper source document, or amount of entry not accurately obtained from source document
R11	Check truncation return
R12	Account sold to another financial institution
R13	Invalid ACH routing number
R14	Representative payee is deceased or cannot continue in that capacity
R15	Beneficiary or account holder other than representative payee deceased
R16	Account funds have been frozen
R17	Item returned because of invalid data; refer to addenda for information

TABLE A-7 eCheck Return Reason Codes

eCheck Return Reason Code	Description
R18	Improper effective date
R19	Amount error
R20	Account does not allow ACH transactions or limit for transactions has been exceeded
R21	Invalid company identification
R22	Invalid individual ID
R23	Credit entry refused by receiver
R24	Duplicate entry
R25	Addenda record error
R26	Mandatory field error
R27	Trace number error
R28	Routing/transit number check digit error
R29	Corporate customer advised not authorized
R30	RDFI not participant in check truncation program
R31	Permissible return entry
R32	RDFI non-settlement
R33	Return of item
R34	Limited participation ODFI
R35	Return of improper debit entry
R36	Return of improper credit entry
R37	Source document presented for payment
R38	Stop payment on source document
R39	Improper source document
R40	Return of item by government agency
R41	invalid Transaction Code
R42	Routing/transit number check digit error
R43	Invalid account number
R44	Invalid individual ID
R45	Invalid individual name or company name

TABLE A-7 eCheck Return Reason Codes

eCheck Return Reason Code	Description
R46	Invalid representative payee indicator code
R47	Duplicate enrollment
R50	State law affecting RCK acceptance
R51	Item is ineligible, notice not provided, signature not genuine, or original item altered for adjustment entry
R52	Stop payment on item
R53	Item and ACH entry presented for payment
R61	Misrouted return - RDFI for original entry has placed incorrect routing/transit number in RDFI identification field
R67	Duplicate return
R68	Untimely return - return was not sent within the established timeframe
R69	Field errors
R70	Permissible return entry not accepted
R71	Misrouted dishonored return -incorrect routing/transit number in RDFI identification field
R72	Untimely return - dishonored return was not sent within the established timeframe
R73	Timely original return - RDFI certifies the original return entry was sent within established timeframe for original returns
R74	Corrected return - RDFI is correcting a previous return entry that was dishonored because it contained incomplete or incorrect information
R75	Original return not a duplicate
R76	No errors found
R80	Cross-border payment coding error
R81	Non-participant in cross-border program
R82	Invalid foreign RDFI identification
R83	Foreign RDFI unable to settle
R84	Cross-border entry not processed by originating gateway operator
R94	Administrative return item was processed and resubmitted as a photocopy

TABLE A-7 eCheck Return Reason Codes

eCheck Return Reason Code	Description
R95	Administrative return item was processed and resubmitted as a MICR-Split
R97	Administrative return item was processed and resubmitted with corrected dollar amount
R98	Indicates a return PAC (pre-authorized check); RDFI provides a text reason and indicated a new account number on the PAC itself
R99	Indicates a return PAC (pre-authorized check); RDFI provides a text reason on the PAC itself for which there is no equivalent return reason code

A.8 eCheck NoC Change Codes

Table A-8 is a list of eCheck NOC Change Codes. These codes are included in the daily NOC report made available to you via sFTP.

TABLE A-8 eCheck NOC Change Codes

eCheck Return Reason Code	Description
C01	Incorrect account number
C02	Incorrect routing/transit number
C03	Incorrect routing/transit number and incorrect account number
C04	Incorrect account name
C05	Incorrect transaction code
C06	Incorrect account number and transaction code
C07	Incorrect routing/transit number, account number and transaction code
C08	Incorrect foreign RDFI identification
C09	Incorrect individual ID
C13	Addenda format error
C61	Misrouted NOC
C62	Incorrect trace number
C63	Incorrect company ID
C64	Incorrect individual ID
C65	Incorrectly formatted correct data
C66	Incorrect discretionary data
C67	Routing/transit number not from original entry
C68	Account number not from original entry
C69	Incorrect transaction code
C96	Administrative return dishonor (dollar amount will be zero)
C99	Converted to MICR draft (check conversion items)



CREDIT CARD NUMBER FORMATS

This appendix has two parts. The first provides basic information about card numbers, such as length, prefixes, and validation numbers. The second part provides information about the Luhn Mod-10 algorithm used to validate account numbers.

Credit Card Number Formats:

[Table B-1](#) provides information on number formats for various credit card types.

CAUTION: The data presented here is for informational purposes only and is subject to change by the Credit Card Associations/Companies. You should verify the information using additional sources prior to using it to create or alter any of your business systems, processes, or procedures.

TABLE B-1 Card Number Formats

Card Type	Card Number Prefix/Range	Number Length	Card Validation Number Length	Comments
American Express	34 and 37	15 digits	4 digits	
Diners Club International	36	14 digits	3 digits	Account Numbers starting with 36 may be submitted as either Discover (recommended) or Diners Club.
Diners Club (US and Canada)	54 and 55	16 digits	3 digits	These are processed through the MasterCard network and must be submitted as MasterCard.

Card Type	Card Number Prefix/Range	Number Length	Card Validation Number Length	Comments
Discover	300000-306000 309500-309600 352800-359000 36 38 39 64 65 6011 622126-622925 624000-262999 628200-628899	14 digits or 16 digits	3 digits	Account Numbers starting with 36 may be submitted as either Discover (recommended) or Diners Club.
JCB	35 (except 352800-358999)	16 digits	3 digits	Account numbers 352800-358999 are processed through the Discover network
MasterCard	51-55	16 digits or 19 digits	3 digits	
Visa	4	16 digits or 19 digits	3 digits	

Luhn Mod-10 Algorithm for Card Number Validation:

The Luhn Mod-10 algorithm was invented in 1954 by IBM scientist Hans Peter Luhn and is a relatively simple formula used in has numerous applications to validate identification numbers, including credit cards. The algorithm detects all single digit errors in an account number, as well as most transpositions of adjacent numbers.

Use the following method to determine if an account number is Mod-10 compliant:

1. Working from the right, double every other number. If the result of any doubling is a 2-digit number, treat them as individual digits for step 2. For example, $2 * 9 = 18$, should be treated as a 1 and an 8.
2. Add all the numbers together, including those you did not double. Remember to treat any 2-digit numbers as individual numbers.

3. If the result of step 2 is a multiple of 10, the account number is Mod-10 compliant.

Example: Mod-10 Algorithm

For the account number 4005550000081019, the computations are shown in the table below.

4	0	0	5	5	5	0	0	0	0	0	8	1	0	1	9
x2		x2		x2		x2		x2		x2		x2		x2	
8	0	0	5	10	5	0	0	0	0	0	8	2	0	2	9
8+	0+	0+	5+	1+0+	5+	0+	0+	0+	0+	0+	8+	2+	0+	2+	9

The result is 40, which is a multiple of 10 and therefore compliant.

Index

Numerics

3DS response certification testing, 92

A

AAVS Response Codes, 450
accNum, 190
accountInformation, 191
accountNumber, 192
accountUpdate, 193
accountUpdateFileRequestData, 194
accountUpdater, 195
accountUpdaterResponse, 199
accType, 200
actionReason, 201
address response certification testing, 87
addressIndicator, 202
addressLine1, 203
Advanced AVS Response Codes, 450
advancedAVSResult, 204
affiliate, 205
affluence, 206
Affluence Indicator Feature, 12
allowPartialAuth, 207
American Express
 Authorization lifespan, 114
amexAggregatorData, 208
amount, 209
approvedAmount, 210
authAmount, 211
authCode, 212
authDate, 213
authenticatedByMerchant, 214
authentication, 215
authenticationResult, 216
authenticationTransactionId, 217
authenticationValue, 218
authInformation, 219
Authorization
 lifespan, 114
authorization, 220
Authorization lifespan
 American Express, 34
 Bill Me Later, 34

Discover, 34
MasterCard, 34
PayPal, 34
Visa, 34
Authorization Reversal transactions
 certification testing, 69
authorizationResponse, 221
authorizationSourcePlatform, 222
authReversal, 223
authReversalResponse, 224
Automatic Account Updater, 26
availableBalance, 225
AVS response code testing, 85
AVS Response Codes, 449
avsResult, 226

B

batchRequest, 227
batchResponse, 231
Bill Me Later
 Authorization lifespan, 114
billMeLaterRequest, 232
billMeLaterResponseData, 234
billToAddress, 235
bin, 237
bmlMerchantId, 238
bmlProductType, 239
bypassVelocityCheck, 240

C

campaign, 241
capability, 242
capture, 243
captureGivenAuth, 244
captureGivenAuthResponse, 245
captureResponse, 246
card, 247
cardAcceptorTaxId, 248
Cardholder Type Indicator Feature, 12
cardholderAuthentication, 249
cardholderId, 250
cardOrToken, 251
cardProductType, 252

cardValidationNum, 253
cardValidationResult, 254
certification testing
 3DS responses, 92
 address responses, 87
 Authorization Reversal transactions, 69
 AVS response codes, 85
 response codes and messages, 90
 volume testing, 103
chargeback, 255
checkNum, 256
city, 257
clinicOtherAmount, 258
code, 259
commodityCode, 260
companyName, 261
country, 262
credit, 263
creditLine, 265
creditResponse, 266
customBilling, 267
customerInfo, 269
customerIpAddress, 270
customerReference, 271
customerRegistrationDate, 272
customerType, 273
customerWorkTelephone, 274

D

data requirements
 Level 2 and Level 3, 303
Decline Notice
 eCheck Verification, 23
deliveryType, 275
dentalAmount, 276
descriptor, 277
destinationCountryCode, 278
destinationPostalCode, 279
detailTax, 280
discountAmount, 281
Discover
 Authorization lifespan, 114
dob, 282
Duplicate Transaction Detection Feature, 6
dutyAmount, 283

E

echeck, 284
eCheck Auto Redeposit Feature, 25
eCheck NoC Update Feature, 25
eCheck Processing Feature, 23
eCheck Validation Feature, 23
eCheck Verification
 Decline Notice, 23
eCheck Verification Feature, 23
eCheckAccountSuffix, 285
echeckCredit, 286
echeckCreditResponse, 287
echeckForToken, 288
echeckOrEcheckToken, 289
echeckRedeposit, 290
echeckRedepositResponse, 291
echeckSale, 292
echeckSaleResponse, 293
echeckToken, 294
echeckVerification, 295
echeckVerificationResponse, 296
echeckVoid, 297
echeckVoidResponse, 298
email, 299
employerName, 300
enhancedAuthResponse, 301
enhancedData, 303
entryMode, 305
expDate, 306
extendedCardResponse, 307

F

Features

Affluence Indicator, 12
Automatic Account Updater, 26
Cardholder Type Indicator, 12
Duplicate Transaction Detection, 6
eCheck Auto Redeposit, 25
eCheck NoC Update, 25
eCheck Processing, 23
eCheck Validation, 23
eCheck Verification, 23
Healthcare Card, 31
International Card Filtering, 14
Issuer Country Indicator, 12
Prepaid Card Filtering, 13

Prepaid Indicator, 11
Report Groups, 9
Smart Authorization, 11
Tokenization, 17
filtering, 308
firstName, 309
forceCapture, 310
forceCaptureResponse, 311
fraudCheck, 391
fraudResult, 312
fundingSource, 313

H

Healthcare Card Feature, 31
healthcareIIAS, 315

I

IIASFlag, 316
incomeAmount, 317
incomeCurrency, 318
international, 319
International Card Filtering Service, 14
invoiceReferenceNumber, 320
Issuer Country Indicator Feature, 12
issuerCountry, 321
itemCategoryCode, 322
itemDescription, 323
itemDiscountAmount, 324
itemSequenceNumber, 325

L

lastName, 326
Level 2 and Level 3 data requirements, 303
lineItemData, 327
lineItemTotal, 329
lineItemTotalWithTax, 330
litleOnlineRequest, 331
litleOnlineResponse, 332
litleRequest, 333
litleResponse, 334
litleSessionId, 335
litleToken, 336
litleTxnId, 337

M

MasterCard
Authorization lifespan, 114
merchantData, 338
merchantGroupingId, 339
merchantId, 340
message, 341
middleInitial, 342

N

name, 343
newAccountInfo, 344
newCardInfo, 345
newCardTokenInfo, 346
newTokenInfo, 347
nextRecycleTime, 348
number, 349

O

optional certification tests
3DS responses, 92
address responses, 87
AVS response codes, 85
response codes and messages, 90
volume testing, 103
orderDate, 350
orderId, 351
orderSource, 352
originalAccountInfo, 353
originalCard, 354
originalCardInfo, 355
originalCardTokenInfo, 356
originalToken, 357
originalTokenInfo, 358

P

password, 359
payerId, 360
paypage, 361
paypageRegistrationId, 362
PayPal
Authorization lifespan, 114
paypal, 363
paypalOrderComplete, 364

phone, 365
pos, 366
postDate, 367
postDay, 368
preapprovalNumber, 369
prepaid, 370
Prepaid Card Filtering Service, 13
Prepaid Indicator Feature, 11
prepaidCardType, 371
processingInstructions, 372
productCode, 373

Q

quantity, 374

R

recycleAdvice, 375
recycleAdviceEnd, 376
recycleBy, 377
recycling, 378
Recycling Engine, 15
Recycling Engine and Auth Reversal, 37, 41
recyclingRequest, 379
registerTokenRequest, 380
registerTokenResponse, 381
reloadable, 382
Report Group Feature, 9
residenceStatus, 383
response, 384
response codes and messages testing, 90
responseTime, 385
RFRRequest, 386
RFRResponse, 387
routingNum, 388
RxAmount, 389

S

sale, 390
saleResponse, 392
salesTax, 393
sellerId, 394
sellerMerchantCategoryCode, 395
shipFromPostalCode, 396
shippingAmount, 397
shipToAddress, 398

Smart Authorization Features, 11
ssn, 399
state, 400
stopping auto-recycling, 37, 41

T

taxAmount, 401
taxExempt, 402
taxIncludedInTotal, 403
taxRate, 404
taxType, 405
taxTypeIdentifier, 406
termsAndConditions, 407
testing
 Authorization Reversal transactions, 69
 optional certification tests
 3DS responses, 92
 address responses, 87
 AVS response codes, 85
 response codes and messages, 90
 volume testing, 103
Timeout settings, 52
token, 408
Tokenization Feature, 17
tokenMessage, 409
tokenResponse, 410
tokenResponseCode, 411
totalHealthcareAmount, 412
track, 413
transactionId, 414
type, 415

U

unitCost, 417
unitOfMeasure, 418
updatedCard, 419
updatedToken, 420
url, 421
user, 422

V

verificationCode, 423
verify, 424
virtualAuthenticationKeyData, 426
virtualAuthenticationKeyPresenceIndicator, 427

Visa

Authorization lifespan, 114
visionAmount, 425
void, 428
voidResponse, 429
volume certification testing, 103

X

XML elements

accNum, 190
accountInformation, 191
accountNumber, 192
accountUpdate, 193
accountUpdateFileRequestData, 194
accountUpdater, 195
accountUpdaterResponse, 199
accType, 200
actionReason, 201
addressIndicator, 202
addressLine1, 203
advancedAVSResult, 204
affiliate, 205
affluence, 206
allowPartialAuth, 207
amexAggregatorData, 208
amount, 209
approvedAmount, 210
authAmount, 211
authCode, 212
authDate, 213
authenticatedByMerchant, 214
authentication, 215
authenticationResult, 216
authenticationTransactionId, 217
authenticationValue, 218
authInformation, 219
authorization, 220
authorizationResponse, 221
authorizationSourcePlatform, 222
authReversal, 223
authReversalResponse, 224
availableBalance, 225
avsResult, 226
batchRequest, 227
batchResponse, 231
billMeLaterRequest, 232
billMeLaterResponseData, 234
billToAddress, 235
bin, 237
bmlMerchantId, 238
bmlProductType, 239
bypassVelocityCheck, 240
campaign, 241
capability, 242
capture, 243
captureGivenAuth, 244
captureGivenAuthResponse, 245
captureResponse, 246
card, 247
cardAcceptorTaxId, 248
cardholderAuthentication, 249
cardholderId, 250
cardOrToken, 251
cardProductType, 252
cardValidationNum, 253
cardValidationResult, 254
chargeback, 255
checkNum, 256
city, 257
clinicOtherAmount, 258
code, 259
commodityCode, 260
companyName, 261
country, 262
credit, 263
creditLine, 265
creditResponse, 266
customBilling, 267
customerInfo, 269
customerIpAddress, 270
customerReference, 271
customerRegistrationDate, 272
customerType, 273
customerWorkTelephone, 274
deliveryType, 275
dentalAmount, 276
descriptor, 277
destinationCountryCode, 278
destinationPostalCode, 279
detailTax, 280
discountAmount, 281
dob, 282
dutyAmount, 283
echeck, 284

eCheckAccountSuffix, 285
echeckCredit, 286
echeckCreditResponse, 287
echeckForToken, 288
echeckOrEcheckToken, 289
echeckRedeposit, 290
echeckRedepositResponse, 291
echeckSale, 292
echeckSaleResponse, 293
echeckToken, 294
echeckVerification, 295
echeckVerificationResponse, 296
echeckVoid, 297
echeckVoidResponse, 298
email, 299
employerName, 300
enhancedAuthResponse, 301
enhancedData, 303
entryMode, 305
expDate, 306
extendedCardResponse, 307
filtering, 308
firstName, 309
forceCapture, 310
forceCaptureResponse, 311
fraudCheck, 391
fraudResult, 312
fundingSource, 313
healthcareIIAS, 315
IIASFlag, 316
incomeAmount, 317
incomeCurrency, 318
international, 319
invoiceReferenceNumber, 320
issuerCountry, 321
itemCategoryCode, 322
itemDescription, 323
itemDiscountAmount, 324
itemSequenceNumber, 325
lastName, 326
lineItemData, 327
lineItemTotal, 329
lineItemTotalWithTax, 330
litleOnlineRequest, 331
litleOnlineResponse, 332
litleRequest, 333
litleResponse, 334
litleSessionId, 335
litleToken, 336
litleTxnId, 337
merchantData, 338
merchantGroupId, 339
merchantId, 340
message, 341
middleInitial, 342
name, 343
newAccountInfo, 344
newCardInfo, 345
newCardTokenInfo, 346
newTokenInfo, 347
nextRecycleTime, 348
number, 349
orderDate, 350
orderId, 351
orderSource, 352
originalAccountInfo, 353
originalCard, 354
originalCardInfo, 355
originalCardTokenInfo, 356
originalToken, 357
originalTokenInfo, 358
password, 359
payerId, 360
paypage, 361
paypageRegistrationId, 362
paypal, 363
paypalOrderComplete, 364
phone, 365
pos, 366
postDate, 367
postDay, 368
preapprovalNumber, 369
prepaid, 370
prepaidCardType, 371
processingInstructions, 372
productCode, 373
quantity, 374
recycleAdvice, 375
recycleAdviceEnd, 376
recycleBy, 377
recycling, 378
recyclingRequest, 379
registerTokenRequest, 380
registerTokenResponse, 381
reloadable, 382
residenceStatus, 383

response, 384
responseTime, 385
RFRRequest, 386
RFRResponse, 387
routingNum, 388
RxAmount, 389
sale, 390
saleResponse, 392
salesTax, 393
sellerId, 394
sellerMerchantCategoryCode, 395
shipFromPostalCode, 396
shippingAmount, 397
shipToAddress, 398
ssn, 399
state, 400
taxAmount, 401
taxExempt, 402
taxIncludedInTotal, 403
taxRate, 404
taxType, 405
taxTypeIdentifier, 406
termsAndConditions, 407
token, 408
tokenMessage, 409
tokenResponse, 410
tokenResponseCode, 411
totalHealthcareAmount, 412
track, 413
transactionId, 414
type, 415
unitCost, 417
unitOfMeasure, 418
updatedCard, 419
updatedToken, 420
url, 421
user, 422
verificationCode, 423
verify, 424
virtualAuthenticationKeyData, 426
virtualAuthenticationKeyPresenceIndicator, 427
visionAmount, 425
void, 428
voidResponse, 429
yearsAtEmployer, 430
yearsAtResidence, 431

zip, 432

Y

yearsAtEmployer, 430
yearsAtResidence, 431

Z

zip, 432