



LittleXML Reference Guide

May 2017

XML Release: 11.0

Document Version: 1.2

Vantiv LittleXML Reference Guide Document Version: 1.2

All information whether text or graphics, contained in this manual is confidential and proprietary information of Vantiv, LLC and is provided to you solely for the purpose of assisting you in using a Vantiv, LLC product. All such information is protected by copyright laws and international treaties. No part of this manual may be reproduced or transmitted in any form or by any means, electronic, mechanical or otherwise for any purpose without the express written permission of Vantiv, LLC. 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 Vantiv, LLC 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 Vantiv, LLC or any other party. Vantiv, LLC 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 © 2003-2017, Vantiv, LLC - ALL RIGHTS RESERVED.

CONTENTS

About This Guide

Intended Audience	xxiii
Revision History	xxiii
Document Structure	xxiv
Documentation Set	xxiv
Typographical Conventions	xxvi
Contact Information.....	xxvii

Chapter 1 Introduction

The LittleXML Data Format.....	2
Communications Protocols	2
General XML Coding Requirements	3
Other XML Resources.....	4
Batch Transaction Processing	5
Recommended Session File Size	5
Payment Integration Platform (LittleXML SDKs).....	6
Duplicate Transaction Detection	7
Batch Duplicate Checking	7
Batch Dupe Checking for Dynamic Payout Funding Instructions.....	8
Online Duplicate Checking.....	8
Coding for Report Groups	10
Additional/Alternate Methods of Tagging Transactions	11
Recovery	12
Authorization/Sale Recycling	12
Recycling Engine	12
Account Updater Service	14
Match Back	15
Merchant Requirements.....	16
Account Updater Features	17
Recurring Engine	18
Payment Plans	18
Subscriptions.....	19
Add Ons and Discounts	21
Recurring Reports	21
Transaction Types and Uses	22
Customer Insight Features.....	25
Prepaid Indicator.....	25
Affluence Indicator	26

Issuer Country Indicator	26
Cardholder Type Indicator.....	26
Flow Control for the Insights Feature	27
Fraud Toolkit	28
Essential Tier	29
Prepaid Card Filtering	29
International BIN Filtering	30
Prior Chargeback Filtering	30
Security Code No-Match Filter	30
Card Velocity Filtering	31
Prior Fraud Advice Filtering	31
AVS Filter	31
Email Velocity Filter.....	32
Phone Velocity Filter	32
IP Velocity Filter	32
Device Velocity Filter.....	33
Application of Filters - Filtering Rules	33
Extended Tier.....	34
Premium Tier.....	35
Modifications to Your Web Page.....	35
LittleXML Transactions.....	36
Information Only Option	38
Fraud Check Transactions	38
Tokenization Feature	39
How Tokenization Works	40
Token Formats	41
Obtaining Tokens	41
Bulk Token Registration	42
Supported Token Transactions	42
Compliance with Visa Best Practices for Tokenization	43
eCheck Processing	45
Validation Feature	45
Verification Feature	45
Required Contents of Decline Notice	46
Automatic Notice of Change (NoC) Updates	47
Auto Redeposit Feature	47
eCheck Prenotification	47
SEPA Direct Debit	49
SEPA Direct Debit - Vantiv Supplied Mandate.....	49
SEPA Direct Debit - Merchant Supplied Mandate.....	52
iDEAL	55

Mobile Point of Sale	58
eCommerce Solution for Apple Pay™	59
Overview of Apple Pay Operation	59
Vantiv Decryption of Apple Pay PKPaymentToken.....	60
Using the Browser JavaScript API for Apple Pay on the Web	60
Using the Vantiv Mobile API for Apple Pay	62
Submitting the Apple Pay PKPaymentToken in LittleXML	63
Merchant Decryption of Apple Pay PKPaymentToken.....	64
Recurring Payments with Apple Pay	66
eCommerce Solution for Android Pay™	67
Android Pay using eProtect.....	67
Merchant Decryption Method	70
Recurring Payments with Android Pay.....	73
Supported Transaction Types	74
Authorization Transaction	74
AVS Only Transaction	75
Authorization Reversal Transactions	75
Notes on the Use of Authorization Reversal Transactions.....	76
Using Authorization Reversal to Halt Recycling Engine.....	77
Activate Transaction.....	77
Activate Reversal Transaction (Online Only)	77
Balance Inquiry Transaction.....	78
Cancel Subscription Transaction	78
Capture Transaction.....	78
Capture Given Auth Transaction.....	78
Create Plan Transaction	79
Credit Transaction.....	79
Deactivate Transaction	80
Deactivate Reversal Transaction (Online Only)	80
Deposit Reversal Transaction (Online Only)	80
eCheck Credit Transaction.....	80
eCheck Prenotification Credit Transaction.....	80
eCheck Prenotification Sale Transaction	80
eCheck Redeposit Transaction	81
eCheck Sales Transaction	81
eCheck Verification Transaction	81
eCheck Void Transaction (Online Only)	81
Force Capture Transaction	81
Gift Card Auth Reversal	82
Gift Card Capture	82
Gift Card Credit	82

Load Transaction	82
Load Reversal Transaction (Online Only)	82
Refund Reversal Transaction (Online Only)	82
Register Token Transaction	82
Sale Transaction	83
Status Query Transaction	83
Unload Transaction	84
Unload Reversal Transaction (Online Only).....	84
Update Card Validation Number Transaction	84
Update Plan Transaction.....	84
Update Subscription Transaction	84
Void Transaction (Online Only)	84
Using Void to Halt Recycling Engine.....	85
Instruction-Based Dynamic Payout Transactions	85

Chapter 2 Testing Your LittleXML Transactions

Certification and Testing Environments	88
Sandbox Environment.....	88
Pre-Live Environment.....	88
Pre-Live Environment Limitations and Maintenance Schedules	89
Post-Live Environment	89
Post-Live Environment Limitations and Maintenance Schedules	89
Overview of Testing	91
Planning for Certification Testing	91
Required Certification Testing.....	92
Optional Testing	92
System Doctor.....	92
Transferring Files	95
Transferring Session Files	95
Submitting a Session File for Processing.....	95
Retrieving Processed Session Files.....	96
Transferring Online Files.....	96
ASP Programming Example	97
Java Programming Example	98
Notes on Timeout Settings.....	99
Notes on Persistent Connections.....	99
Helpful Web Sites.....	100
Performing the Required Certification Tests	101
Testing Authorization (including Indicators), AVS Only, Capture, Credit, Sale, and Void Transactions.....	101
Testing Authorization Reversal Transactions.....	114

Testing eCheck Transactions.....	117
Testing Token Transactions.....	124
Testing Query Transactions	129
Performing the Optional Tests	131
Testing AVS and Card Validation.....	132
Testing Address Responses	133
Testing Advanced AVS Response Codes.....	135
Testing Response Reason Codes and Messages	136
Testing 3DS Responses	139
Testing the Prepaid Filtering Feature.....	141
Testing the International Card Filter Feature	143
Testing Security Code No-Match Filtering	144
Testing Advanced Fraud Tools	146
Testing Account Updater.....	148
Testing Account Updater Extended Response Codes	149
Testing Account Updater for Tokenized Merchants	150
Testing Tax Billing.....	150
Testing Convenience Fees	151
Testing the Recycling Engine.....	153
Testing Recycling Engine Cancellation	164
Testing Recurring Engine Transactions	166
Testing Gift Card Transactions	179
Testing Mobile Point of Sale	189
Testing MasterPass Transactions.....	191
Testing Apple Pay Transaction Processing	192
Testing the Submission of the Decrypted PKPaymentToken in LittleXML.....	192
Testing the Submission of PKPaymentToken in LittleXML	193
Testing Android Pay Transaction Processing	194
Testing Android Pay using eProtect.....	194
Testing SEPA Direct Debit Transaction	196
Testing iDEAL Transactions.....	198
Testing Online Duplicate Transaction Processing	199
Testing Transaction Volume Capacity	200

Chapter 3 LittleXML Transaction Examples

Overview of Online and Batch Processing Formats	202
Batch Process Format.....	202
Supported Communication Protocols.....	203
Batch Processing Request Format	203
Batch Processing Response Format.....	203
Online Processing Format	204

Supported Communication Protocols.....	205
Online Processing Request Format	205
Online Processing Response Format	205
Transaction Types and Examples.....	206
Authorization Transactions.....	208
Authorization Request Structure	208
Authorization Response Structure	215
Authorization Reversal Transactions	220
Authorization Reversal Requests.....	220
Authorization Reversal Responses	222
Activate Transactions.....	223
Activate Request.....	223
Activate Response	224
Activate Reversal Transactions (Online Only)	226
Activate Reversal Request.....	226
Activate Reversal Response	227
Balance Inquiry Transactions.....	228
Balance Inquiry Request	228
Balance Inquiry Response	229
Cancel Subscription Transactions.....	230
Cancel Subscription Request.....	230
Cancel Subscription Response	230
Capture Transactions.....	231
Capture Request.....	231
Capture Response	238
Capture Given Auth Transactions	240
Capture Given Auth Request	240
Capture Given Auth Response	244
Create Plan Transactions.....	246
Create Plan Request.....	246
Create Plan Response	247
Credit Transactions	247
Credit Request for a Vantiv Processed Transaction	248
Credit Request for a Non-Vantiv Processed Transaction	251
Credit Response	256
Deactivate Transactions.....	257
Deactivate Request.....	257
Deactivate Response	258
Deactivate Reversal Transactions (Online Only)	259
Deactivate Reversal Request.....	259
Deactivate Reversal Response	260

Deposit Reversal Transactions (Online Only)	261
Deposit Reversal Request	261
Deposit Reversal Response.....	262
eCheck Credit Transactions.....	263
eCheck Credit Request Against a Vantiv Transaction	263
eCheck Credit Request for a Non-Vantiv Processed Sale.....	265
eCheck Credit Response	266
eCheck Prenotification Credit Transactions (Batch Only)	266
eCheck Prenotification Credit Request	267
eCheck Prenotification Credit Response	268
eCheck Prenotification Sale Transactions (Batch Only)	269
eCheck Prenotification Sale Request.....	269
eCheck Prenotification Sale Response	270
eCheck Redeposit Transactions	271
eCheck Redeposit Request	271
eCheck Redeposit Response.....	273
eCheck Sale Transactions	274
eCheck Sale Request	274
eCheck Sale Response.....	276
eCheck Verification Transactions.....	277
eCheck Verification Request.....	278
eCheck Verification Response	280
eCheck Void Transactions (Online Only)	281
eCheck Void Request	281
eCheck Void Response.....	282
Force Capture Transactions.....	283
Force Capture Request.....	283
Force Capture Response	286
Fraud Check Transaction.....	288
Fraud Check Request	288
Fraud Check Response	289
Gift Card Auth Reversal Transactions.....	290
Gift Card Auth Reversal Request.....	290
Gift Card Auth Reversal Response	291
Gift Card Capture Transactions	292
Gift Card Capture Request.....	292
Gift Card Capture Response	293
Gift Card Credit Transactions.....	294
Gift Card Capture Request.....	294
Gift Card Credit Response	295
Load Transactions.....	296

Load Request.....	297
Load Response	297
Load Reversal Transactions (Online Only)	298
Load Reversal Request.....	299
Load Reversal Response	300
Status Query Transactions (Online Only)	300
Query Transaction Request	301
Query Transaction Response	301
Refund Reversal Transactions (Online Only).....	305
Refund Reversal Request.....	305
Refund Reversal Response	306
Register Token Transactions	307
Register Token Request	307
Register Token Response.....	310
RFR Transactions (Batch Only)	312
RFR Request	312
RFR Response.....	313
Sale Transactions	314
Sale Request.....	314
Sale Response.....	319
Unload Transactions	325
Unload Request	325
Unload Response.....	326
Unload Reversal Transactions (Online Only).....	327
Unload Reversal Request	327
Unload Reversal Response	328
Update Plan Transactions.....	329
Update Plan Request	329
Update Plan Response	329
Update Subscription Transactions	330
Update Subscription Request	330
Update Subscription Response.....	332
Update Card Validation Number Transactions.....	332
Update Card Validation Number Request.....	333
Update Card Validation Number Response	333
Void Transactions (Online Only)	334
Void Request.....	335
Void Response.....	335
 Chapter 4 LittleXML Elements	
accNum.....	338

accountInfo	339
accountInformation	340
accountNumber.....	341
accountNumberLength	342
accountUpdate	343
accountUpdateFilerequestData	344
accountUpdater.....	345
accountUpdateResponse.....	349
accType	350
actionReason	351
activate	352
activateResponse	353
activateReversal	354
activateReversalResponse	355
active	356
addOnCode	357
addressIndicator	358
addressLine1, addressLine2, addressLine3	359
advancedAVSResult	360
advancedFraudChecks	361
advancedFraudResults	362
affiliate.....	363
affluence	364
allowPartialAuth	365
amexAggregatorData.....	366
amount	367
androidpayResponse	369
applepay	370
applepayResponse	371
applicationData	372
applicationExpirationDate	373
applicationPrimaryAccountNumber.....	374
approvedAmount.....	375
authAmount.....	376
authCode	377
authDate	378
authenticatedByMerchant	379
authentication.....	380
authenticationResult	381
authenticationTransactionId	382
authenticationValue	383

authInformation	384
authorization	385
authorizationResponse	386
authorizationSourcePlatform.....	387
authReversal.....	388
authReversalResponse.....	389
availableBalance.....	390
avsResult	391
balanceInquiry	392
balanceInquiryResponse	393
batchRequest.....	394
batchResponse	402
beginningBalance	404
billingDate	405
billMeLaterRequest	406
billMeLaterresponseData.....	408
billToAddress	409
bin	411
bmlMerchantId	412
bmlProductType.....	413
bypassVelocityCheck	414
campaign	415
cancelSubscription	416
cancelSubscriptionResponse	417
capability	418
capture	419
captureAmount.....	420
captureGivenAuth	421
captureGivenAuthResponse	422
captureResponse.....	423
card	424
cardAcceptorTaxId.....	426
cardholderAuthentication	427
cardholderId	428
cardholderName	429
cardOrToken	430
cardProductType.....	431
cardSuffix	432
cardValidationNum.....	433
cardValidationResult	434
cashBackAmount	435

catLevel	436
ccdPaymentInformation	437
chargeback	438
checkNum	439
city	440
clinicOtherAmount	441
code	442
commodityCode	443
companyName	444
country	445
createAddOn	446
createDiscount	447
createPlan	448
createPlanResponse	449
credit	450
creditAmount	451
creditLine	452
creditLittleTxnId	453
creditResponse	454
cryptogram	455
currencyCode	456
customAttribute1	457
customBilling	458
customIdentifier	460
customerInfo	461
customerIpAddress	462
customerReference	463
customerRegistrationDate	464
customerType	465
customerWorkTelephone	466
data	467
deactivate	468
deactivateResponse	469
deactivateReversal	470
deactivateReversalResponse	471
debtRepayment	472
deleteAddOn	473
deleteDiscount	474
deliveryType	475
dentalAmount	476
depositReversal	477

depositReversalResponse	478
description	479
descriptor	480
destinationCountryCode	481
destinationPostalCode	482
detailTax	483
deviceManufacturerIdentifier.....	484
deviceReputationScore	485
deviceReviewStatus.....	486
discountAmount	487
discountCode	488
dob	489
dutyAmount.....	490
echeck.....	491
eCheckAccountSuffix	492
echeckCredit	493
echeckCreditResponse	494
echeckForToken	495
echeckOrEcheckToken.....	496
echeckPreNoteCredit	497
echeckPreNoteCreditResponse	498
echeckPreNoteSale	499
echeckPreNoteSaleResponse	500
echeckRedeposit	501
echeckRedepositResponse	502
echeckSale	503
echeckSalesResponse	504
echeckToken.....	505
echeckVerification.....	506
echeckVerificationResponse.....	507
echeckVoid	508
echeckVoidResponse	509
ecilIndicator.....	510
email	511
employerName.....	512
encryptedTrack	513
endDate	514
endingBalance	515
endpoint	516
enhancedAuthResponse.....	517
enhancedData.....	519

entryMode	523
ephemeralPublicKey	524
expDate	525
expMonth	526
expYear	527
extendedCardResponse	528
fieldValue	529
filtering	530
finalPayment	531
firstName	532
forceCapture	533
forceCaptureResponse	534
formatId	535
fraudCheck	536
fraudCheckResponse	537
fraudFilterOverride	538
fraudResult	539
fundingInstructionVoid	540
fundingInstructionVoidResponse	541
fundingSource	542
fundingSubmerchantId	543
fundsTransferId	544
giftCardAuthReversal	545
giftCardAuthReversalResponse	546
giftCardBin	547
giftCardCapture	548
giftCardCaptureResponse	549
giftCardCredit	550
giftCardCreditResponse	551
giftCardResponse	552
giropay	553
giropayResponse	554
header	555
healthcareAmounts	556
healthcareIIAS	557
iban	558
ideal	559
idealResponse	560
IIASFlag	561
incomeAmount	562
incomeCurrency	563

international	565
intervalType	566
invoiceReferenceNumber	567
issuerCountry	568
itemCategoryCode	569
itemDescription	570
itemDiscountAmount	571
itemSequenceNumber	572
ksn	573
lastName	574
lineItemData	575
lineItemTotal	577
lineItemTotalWithTax	578
littleInternalRecurringRequest	579
littleOnlineRequest	580
littleOnlineResponse	581
littleRequest	583
littleResponse	584
littleSessionId	586
littleToken	587
littleTxnId	588
load	590
loadResponse	591
loadReversal	592
loadReversalResponse	593
mandateProvider	594
mandateReference	595
mandateSignatureDate	596
mandateURL	597
matchCount	598
merchantData	599
merchantGroupingId	600
merchantId	601
message	602
middleInitial	603
mpos	604
name	605
networkField	606
networkResponse	608
networkSubField	609
networkTransactionId	610

newAccountInfo	611
newCardInfo	612
newCardTokenInfo	613
newTokenInfo	614
nextRecycleTime	615
number.....	616
numberOfPayments	617
onlinePaymentCryptogram	618
orderDate	619
orderId.....	620
orderSource	621
originalAccountInfo	623
origAccountNumber	624
origActionType	625
origId	627
originalAmount	628
originalCard.....	629
originalCardInfo	630
originalCardTokenInfo	631
originalNetworkTransactionId	632
originalRefCode	633
originalSequenceNumber	634
originalSystemTracId	635
originalToken	636
originalTokenInfo	637
originalTransactionAmount	638
originalTxnTime	639
origLittleTxnId	640
origOrderId	641
password	642
payerId	643
payFacCredit	644
payFacCreditResponse	645
payFacDebit	646
payFacDebitResponse	647
paymentDataType	648
paymentPurpose.....	649
paypage	650
paypageRegistrationId	651
paypal	652
payPalNotes	653

payPalOrderComplete	654
phone	655
phone as a child of billToAddress and shipToAddress	655
phone as a child of customBilling	655
physicalCheckCredit	656
physicalCheckCreditResponse	657
physicalCheckDebit	658
physicalCheckDebitResponse	659
pin	660
planCode	661
pos	662
postDate	663
postDay	664
preapprovalNumber	665
preferredLanguage	666
prepaid	667
prepaidCardType	668
processingInstructions	669
processingType	670
productCode	671
publicKeyHash	672
quantity	673
queryTransaction	674
queryTransactionResponse	675
queryTransactionUnavailableResponse	676
recurringRequest	677
recurringResponse	678
recurringTxnId	679
recycleAdvice	680
recycleAdviceEnd	681
recycleBy	682
recycleEngineActive	683
recycleId	684
recycling	685
recycling Element as a Child of authorizationResponse or saleResponse	685
recycling Element as a Child of voidResponse	685
recyclingRequest	687
redirectToken	688
redirectUrl	689
refCode	690
refundReversal	691

refundReversalResponse	692
registerTokenRequest.....	693
registerTokenResponse.....	694
reloadable	695
reserveCredit	696
reserveCreditResponse	697
reserveDebit	698
reserveDebitResponse	699
residenceStatus	700
response	701
responseCode	702
responseMessage	703
responseTime	704
results_Max10	705
RFRRequest	708
RFRResponse	709
routingNum	710
RxAmount	711
sale	712
saleResponse	713
salesTax.....	714
secondaryAmount	715
sellerId	716
sellerMerchantCategoryCode	717
sepaDirectDebit	718
sepaDirectDebitResponse	719
sequenceNumber.....	720
sequenceType	721
shipFromPostalCode	722
shippingAmount	723
shipToAddress	724
signature	725
sofort	726
sofortResponse	727
ssn	728
startDate	729
state	730
submerchantCredit	731
submerchantCreditResponse	732
submerchantDebit	733
submerchantDebitResponse	734

submerchantName.....	735
subscription	736
subscriptionId	738
surchargeAmount	739
systemTracId	740
taxAmount.....	741
taxExempt.....	742
taxIncludedInTotal.....	743
taxRate.....	744
taxType	745
taxTypeIdentifier	746
terminalId	747
termsAndConditions.....	748
threatMetrixSessionId	749
token	750
token (Vantiv generated card number replacement).....	750
token (PayPal generated)	750
tokenMessage.....	751
tokenResponse	752
tokenResponseCode	753
totalHealthcareAmount	754
track	755
track1Status	756
track2Status	757
transactionAmount	758
transactionId	759
transactionId as a Child of the paypal element	759
transactionId as a Child of the header element.....	759
trialIntervalType	761
trialNumberofIntervals	762
triggeredRule	763
txnTime	764
type	765
type Element as a Child of the parent elements listed below.....	765
type Element as a Child of fundingSource	766
unitCost.....	767
unitOfMeasure	768
unload	769
unloadResponse	770
unloadReversal	771
unloadReversalResponse	772

updateAddOn	773
updatedCard	774
updateCardValidationNumOnToken	775
updateCardValidationNumOnTokenResponse	776
updateDiscount	777
updatePlan	778
updatePlanResponse	779
updateSubscription	780
updateSubscriptionResponse	784
updatedToken	785
url	786
user	787
vendorCredit	788
vendorCreditResponse	789
vendorDebit	790
vendorDebitResponse	791
vendorName	792
verificationCode	793
verify	794
version	795
virtualAccountNumber	796
virtualAuthenticationKeyData	797
virtualAuthenticationKeyPresenceIndicator	798
virtualGiftCard	799
virtualGiftCardBin	800
virtualGiftCardResponse	801
visionAmount	802
void	803
voidResponse	804
wallet	805
walletSourceType	806
walletSourceTypeld	807
yearsAtEmployer	808
yearsAtResidence	809
zip	810

Appendix A Payment Transaction Response Codes

Payment Transaction Response Codes	812
3DS Authentication Result Codes	835
AVS Response Codes	837
AAVS Response Codes	838

Card Validation Response Codes	841
Advanced Fraud Tools Triggered Rules	842
XML Validation Error Messages	859
Additional Response Header Error Messages	861
ACH Return Reason Codes	863
ACH NoC Change Codes	867
Canadian eCheck Return Codes	868

Appendix B Credit Card Number Formats

Appendix C Test Card Numbers

Appendix D PayFac™ Dynamic Payout

Advantages of Using Dynamic Payout.....	880
Overview of Dynamic Payout.....	881
Timing of Transactions, Reports, and Money Movement.....	882
Money Movement and Accounts.....	883
Account Balance Verifications.....	884
Example of Funding Instructions.....	885
Funding Instruction Void Transactions.....	888
Funding Instruction Certification Testing.....	889
SSR Reports	891
Tax ID Validation Process.....	893

ABOUT THIS GUIDE

This manual serves as a reference to the LittleXML transaction formats used for payment processing with Vantiv, LLC. It also explains how to perform unattended transaction testing and attended certification testing with Vantiv.

Intended Audience

This document is intended for technical personnel who will be setting up and maintaining payment processing using the LittleXML 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 LittleXML Reference Guide for schema V11.0, including new Gift Card transactions and international alternate payment methods.	N/A
1.1	Added <customIdentifier> to several eCheck transactions. Added information about the iDEAL international payment method, which is now GA.	Chapters 3 and 4 Chapters 1, 2, and 4
1.2	Corrected maintenance window times for Pre-/Post-Live. Added "Appears in Declined Transaction report." note to certain Response Codes.	Chapter 2 Appendix A

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 Vantiv 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.

Appendix C, "Test Card Numbers"

This appendix provides credit card number that can be used for testing.

Appendix D, "PayFac™ Dynamic Payout"

This appendix provides information about PayFac Dynamic Payout transactions.

Documentation Set

For additional information concerning the Vantiv application, see any of the following guides in the documentation set:

- *iQ Reporting and Analytics User Guide*
- *Vantiv Chargeback API Reference Guide*
- *Vantiv Chargeback Process Guide*
- *Vantiv eCommerce Partner Integration Overview Guide*
- *Vantiv PayPal Integration Guide*

- *Vantiv PayFac API Reference Guide*
- *Vantiv PayFac Portal User Guide*
- *Vantiv PayFac Integration Overview Guide*
- *Vantiv eProtect™ Integration Guide*
- *Vantiv Enterprise eProtect™ Integration Guide*
- *Vantiv XML Differences Guide*
- *Vantiv Secure Scheduled Reports Reference 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).
monospaced text	Used in code examples and elsewhere to designate field/element names.

Contact Information

This section provides contact information for organizations within Vantiv.

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

Implementation Contact Information

E-mail	implementation@vantiv.com
Hours Available	Monday – Friday, 8:30 A.M.– 5:30 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	1-844-843-6111 (option 4)
E-mail	chargebacks@vantiv.com
Hours Available	Monday – Friday, 7:30 A.M.– 5:00 P.M. EST

Technical Support - For technical issues such as file transmission errors, email Technical Support. A Technical Support Representative will contact you within 15 minutes to resolve the problem.

Technical Support Contact Information

E-mail	eCommerceSupport@vantiv.com
Hours Available	24/7 (seven days a week, 24 hours a day)

Relationship Management/Customer Service - For non-technical issues, including questions concerning iQ Reporting and Analytics, help with passwords, modifying merchant details, and changes to user account permissions, contact the Relationship Management/Customer Service Department. If you are a Payment Facilitator (PayFac), refer to the second table.

Relationship Management/Customer Service Contact Information - Merchants

Telephone	1-844-843-6111 (Option 3)
E-mail	ecc@vantiv.com
Hours Available	Monday – Friday, 8:00 A.M.– 6:00 P.M. EST

Relationship Management/Customer Service Contact Information - Payment Facilitators

Telephone	1-844-843-6111 (Option 5)
E-mail	PayFacEComm@vantiv.com
Hours Available	Monday – Friday, 8:00 A.M.– 5:00 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@vantiv.com
---------------	--

INTRODUCTION

The LittleXML 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. Typically, merchants use the Online method for Authorization transactions, as well as transactions available only via Online (for example, Void). The Batch method enables you to submit multiple transactions simultaneously in a single Session file. Vantiv recommends the Batch method for all transaction submissions except Authorizations and the transaction available only via Online.

This chapter provides an overview of the LittleXML data format, including some basic XML coding requirements. Also discussed are the advantages of using batch processing, duplicate transaction detection, report groups, Value Added Services, and supported transaction types.

The topics discussed in this chapter are:

- [The LittleXML Data Format](#)
- [Batch Transaction Processing](#)
- [Payment Integration Platform \(LittleXML SDKs\)](#)
- [Duplicate Transaction Detection](#)
- [Coding for Report Groups](#)
- [Recovery](#)
- [Recurring Engine](#)
- [Customer Insight Features](#)
- [Fraud Toolkit](#)
- [Tokenization Feature](#)
- [eCheck Processing](#)
- [SEPA Direct Debit](#)
- [iDEAL](#)
- [Mobile Point of Sale](#)
- [eCommerce Solution for Apple Pay™](#)
- [eCommerce Solution for Android Pay™](#)
- [Supported Transaction Types](#)

1.1 The LittleXML Data Format

There are several advantages to using the LittleXML format 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 Vantiv'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 LittleXML format allows you to explicitly tie deposits to their associated authorizations via the `<littleTxnId>` 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 LittleXML format associates related transactions using the `<littleTxnId>` 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, Vantiv automatically pulls the information from the original transaction. Most other formats require you to resubmit the related data with each transaction.
- **Superior Reporting** - The LittleXML format allows you to separate your transactions into different categories by specifying a Report Group on each transaction. When accessing your data on the iQ reporting and Analytics Interface, 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 LittleXML format explicitly ties related transactions, allowing you and Vantiv to see authorization-to-deposit and deposit-to-refund relationships with precision. This knowledge is indispensable when fighting chargebacks.
- **New Features** - New features developed for the Vantiv eCommerce platform are first exposed via the XML API. While the SDKs add new capabilities shortly after development, direct coding to the XML API gains access to these feature and capabilities earlier.

1.1.1 Communications Protocols

There are two communication protocols supported for the submission of Batch transactions to Vantiv eCommerce platform for processing and one for Online submissions as shown in [Table 1-1](#). For Batch submissions Vantiv 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	N/A	Required
TCP/IP Socket	SSL	N/A	N/A
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 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 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, Vantiv 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.

Be sure to review data provided by customers for special character handling. For example, an address of "4th & Main," must be rewritten as "4th & Main" (including quotes) before being submitted via XML. Failure to quote this type of input causes rejection of XML submissions due to syntax errors.

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>, referred to as a Session, can contain many batches and each batch can contain multiple transactions. We recommend 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** - We optimize 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.

1.2.1 Recommended Session File Size

As stated above, a Session file is a group of batches. A batch is a set of transactions for a single merchant. Normally, you send in a single file which has one batch for each merchant. This works well when the overall number of transactions is small. The number of transactions you should submit in any individual Session or Batch depend on a number of factors, including whether or not you are an individual merchant or a presenter submitting transactions for multiple merchants. In general, you should keep the following recommendations and rules in mind when determining the number of transactions you submit in an Session/Batch file:

- A Batch should not exceed 20,000 transactions. If the number of transaction for a single merchant exceeds 20,000, you should create multiple batches for the same merchant, each batch containing not more than 20,000 transactions.
- A Batch should not contain only one transaction, unless the merchant has only one transaction for the day.
- A Session file must never contain more than 9,999 Batches.
- A Session file must never contain more than 1,000,000 transactions across all Batches.
- Always allow sufficient time between your submission time and your cut-off time for the processing of the Session. Larger files take longer to process.

1.3 Payment Integration Platform (LittleXML SDKs)

In order to facilitate integration to the platform, Vantiv provides several language specific SDKs (Software Development Kits). The developers page on our website (<http://www.vantiv.com/developers>) provides links to SDK libraries for several popular languages, including:

- PHP
- Ruby
- Java
- .NET
- Python

Extensions for:

- Magento
- Opencart

In addition to the SDKs and extensions, Vantiv provides examples of each supported transaction type, as well as demonstration applications. Once you install the library appropriate to your language, the Vantiv Sandbox, which functions as an emulator of our production environment, is available to validate your transaction format.

1.4 Duplicate Transaction Detection

In order to help you avoid transaction errors, Vantiv performs duplicate transaction checking for both Online and Batch transaction submissions. While we use a robust duplicate checking methodology, we cannot guarantee our system will catch all duplicates and bear no responsibility for correcting the impact of erroneously submitted transactions. 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.

For transactions submitted using other formats (e.g., PTI, Nabanco, etc.), the logic used to detect duplicates for these supported, but foreign APIs, is less robust and may miss duplicates in certain scenarios. Vantiv recommends, for better dupe checking, convert to the LittleXML format, as soon as possible.

1.4.1 Batch Duplicate Checking

When processing a Batch, the 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.

NOTE: For information about duplicate checking of Dynamic Payout Funding Instructions see [Batch Dupe Checking for Dynamic Payout Funding Instructions on page 8](#).

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 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.

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 must be processed completely to be included in the previous five days of data. For example, if multiple submitted Batches 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.

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 as a duplicate. When either threshold is met, Vantiv will not process the Batch. If neither threshold is met, Vantiv continues processing the Batch, including any transactions that may have been duplicates.

1.4.1.1 Batch Dupe Checking for Dynamic Payout Funding Instructions

Duplicate checking for funding instructions takes place at the Batch level. The system compares a submitted Batch of instructions to other Batches submitted and accepted on the same day. If a submitted Batch has the same totals and counts as a Batch accepted on the same day, the system rejects the new Batch as a duplicate. If you resubmit a previously rejected Batch, it will not fail dupe checking, because the initial submission was not accepted and is not included in dupe checking comparisons.

NOTE: **If you believe we rejected a funding instruction Batch in error, please contact your Partner Account Manager. If necessary, we can disable the dupe check feature for a particular Batch.**

1.4.2 Online Duplicate Checking

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

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.

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 system.

If you elect to submit Online Sale transactions, Vantiv 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, The duplicate transactions appears in the Declined Transaction report with a Response Reason Code of 251 - Duplicate Transaction. You can access this report in Vantiv iQ or via the Vantiv Secure Scheduled Report. The iQ version provides information in near real-time, while the SSR version runs daily, providing information for the transaction submitted the previous day.

NOTE: If you do not receive a response for a submitted transaction, Vantiv recommends you use the queryTransaction to determine the status of the original transaction (see [Status Query Transactions \(Online Only\)](#) on page 300)

1.5 Coding for Report Groups

You use Report Groups (`reportGroup` attribute) to separate your transactions 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.

CAUTION: Creation of an excessive number of Report Groups (in excess of 250) will impact the amount of time required to compile various reports in the iQ. Report Groups are intended to allow you to segregate transactions by logically grouping them into the different segments of your business.

Report Groups are not intended to track sales or marketing programs that exist for limited times. For this type of tracking, Vantiv provides other transaction tagging methods detailed in [Additional/Alternate Methods of Tagging Transactions](#) on page 11.

Example: Report Groups

The merchant, Demo, wants to separate their domestic and international sales information. To do this the company submits all domestic transactions using `reportGroup = "Domestic Business"`, and all international transactions using `reportGroup = "International Business"`. When they access the Authorization Report in the iQ using the By Reporting Group tab, the transactions would be separated as shown in [Figure 1-1](#).

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

FIGURE 1-1 Report Group Example - 2 Groups

By Reporting Group	By Activity Date	By Payment Method	By Reason	All Currencies	By Presenter	Recycling Summary	Recycling By Attempt
Reporting Group ▾							
<input checked="" type="checkbox"/> Demo				Total Attempts	Declined Auths	Declined %	Approved Auths
<input checked="" type="checkbox"/> Domestic Business				55	21	38.18%	34
International Business				53	21	39.62%	32
Totals:				2	0	0.00%	2
				55	21	38.18%	34
						Approved %	61.82%

The plus sign next to the Domestic Business report group signifies that there are child groups present. When fully expanded (see [Figure 1-2](#)), the iQ shows a report group hierarchy with information for the Domestic Business group further separated into Service A and Service B groups and Service A containing two additional child groups. If you find it necessary to establish this type of nested hierarchy, your Implementation Consultant will assist you.

FIGURE 1-2 Report Group Example - Expanded to Show Child Groups

By Reporting Group	By Activity Date	By Payment Method	By Reason	All Currencies	By Presenter	Recycling Summary	Recycling By Attempt
Reporting Group ▾		Total Attempts	Declined Auths	Declined %	Approved Auths	Approved %	
<input type="checkbox"/> Demo		55	21	38.18%	34	61.82%	
<input type="checkbox"/> Domestic Business		53	21	39.62%	32	60.38%	
<input type="checkbox"/> Service A		23	14	60.87%	9	39.13%	
Product 1		19	14	73.68%	5	26.32%	
Product 2		4	0	0.00%	4	100.00%	
Service B		30	7	23.33%	23	76.67%	
International Business		2	0	0.00%	2	100.00%	
Totals:		55	21	38.18%	34	61.82%	

1.5.1 Additional/Alternate Methods of Tagging Transactions

If you are using schema version 7.x or above you can use the `merchantData` element and its children to tag transactions (Authorization, Sale, Credit, Force Capture, Capture Given Auth, eCheck Sale, and eCheck Credit) with additional information. The three children of `merchantData`: `campaign`, `affiliate`, and `merchantGroupId`, allow you to designate transactions as members of different groups enabling a deeper analysis of sales patterns.

NOTE:

The `merchantData` element and its children were add to the schema in V8.8 and backported to V7.3. If you are using a schema version between 7.0 and 8.7, you can code for the use of these elements and still pass the LittleXML validation.

For example, if the merchant from the previous example were trying a new sales initiative for Product 2 during the month of September. They plan to run ads in Boston and New York to test the new offering. To allow a deeper analysis of sales resulting from the new campaign, they can add the `campaign` element with a value of "September Ads" to the transactions originating in both test market. They can also include the `merchantgroupId` with values that reflect the city where the order originates. By exporting either the Session report or the NSS by Transaction report from the iQ, the company can sort their sales data based upon these fields and gain a better understanding of the effectiveness of the sales campaign.

NOTE:

The transaction tagging elements described above appear in the exported Session and NSS by Transaction reports. They are also visible within the iQ in the new Transaction Detail reports.

1.6 Recovery

Recovery is a bundle of services that include both Account Updater and Recycling Engine. By combining these two managed services into a single bundle, Vantiv simultaneously increases your approval rates, optimizes customer lifetime value, and improves your cash flow, while reducing the cost of implementing the individual features separately in terms of IT resources. For additional information about the capabilities included in this bundle, please refer to [Recycling Engine](#) on page 12, and (AAU) [Account Updater Service](#) on page 14.

1.6.1 Authorization/Sale 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, should devise a strategy for dealing with declined authorizations. As part of optimizing their operations, merchants must devise plans for both the timing 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.

1.6.1.1 Recycling Engine

The 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 Vantiv service removes the need to plan your own recycling strategy.

Recycling Engine has 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

In order to determine the most effective recycle timing, Vantiv 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 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, per Visa regulations.**
For MasterCard and Discover transactions, we retry declined Authorizations a maximum of 8 times within 28 days.

Vantiv provides the results of the recycling efforts to you in a Batch posted daily to your Vantiv sFTP account. 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, Vantiv 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.

Declined Transactions Not Recycled

The system does not recycle transactions declined with one of the Response Codes listed in [Table 1-3](#) below. The <recycleEngineActive> element in the response files indicates if the transaction is being handled by the Recycling Engine.

TABLE 1-3 Response Codes Not Recycled

Response Code	Message
213	Pickup Card - Lost Card
214	Pickup Card - Stolen Card
303	Pick up Card ¹
304	Lost/Stolen Card ¹
305	Expired Card ¹
308	Restricted Card - Chargeback
309	Restricted Card - Prepaid Card Filtering Service
312	Restricted Card - International Card Filtering Service
315	Restricted Card - Auth Fraud Velocity Filtering Service
318	Restricted Card - Auth Fraud Advice Filtering Service
323	No such Issuer ¹
328	Cardholder requested that recurring or installment payment be stopped
358	Restricted by Little due to security code mismatch
550	Restricted Device or IP - ThreatMetrix Fraud Score Below Threshold

1. Not recycled for MasterCard, unless updated account information is available.

Transaction Signature

The Recycling Engine analyzes each Authorization or Sale request message to determine if it is a new request. The result of the analysis determines if the transaction should be added to the recycling pool upon decline or if the system should intercept the transaction to prevent a duplicate transaction entering the recycling pool. To perform the analysis, the system checks the transaction signature. Depending upon your configuration, the transaction signature can be:

- Value of the <recycleId> element
- Value of the <orderId> element
- Values of the <orderId>, <number>, and <amount> elements

NOTE: If you submit a transaction with the identical signature, but containing new information (for example, a new card number), the system updates the transaction in the recycling pool with the new info and continues to recycle.

Additional Configuration Options

The Recycling Engine allows you the additional flexibility of excluding certain transactions from automatic recycling. You can exclude transactions manually by including the <recycleBy> element set to **None**. There are also global controls that allow you to exclude transactions based upon either submission by a particular presenter, or based upon the transaction type (authorization or sale). Please consult your Relationship Manager about the global options, since they must be configured in your Vantiv Merchant Profile.

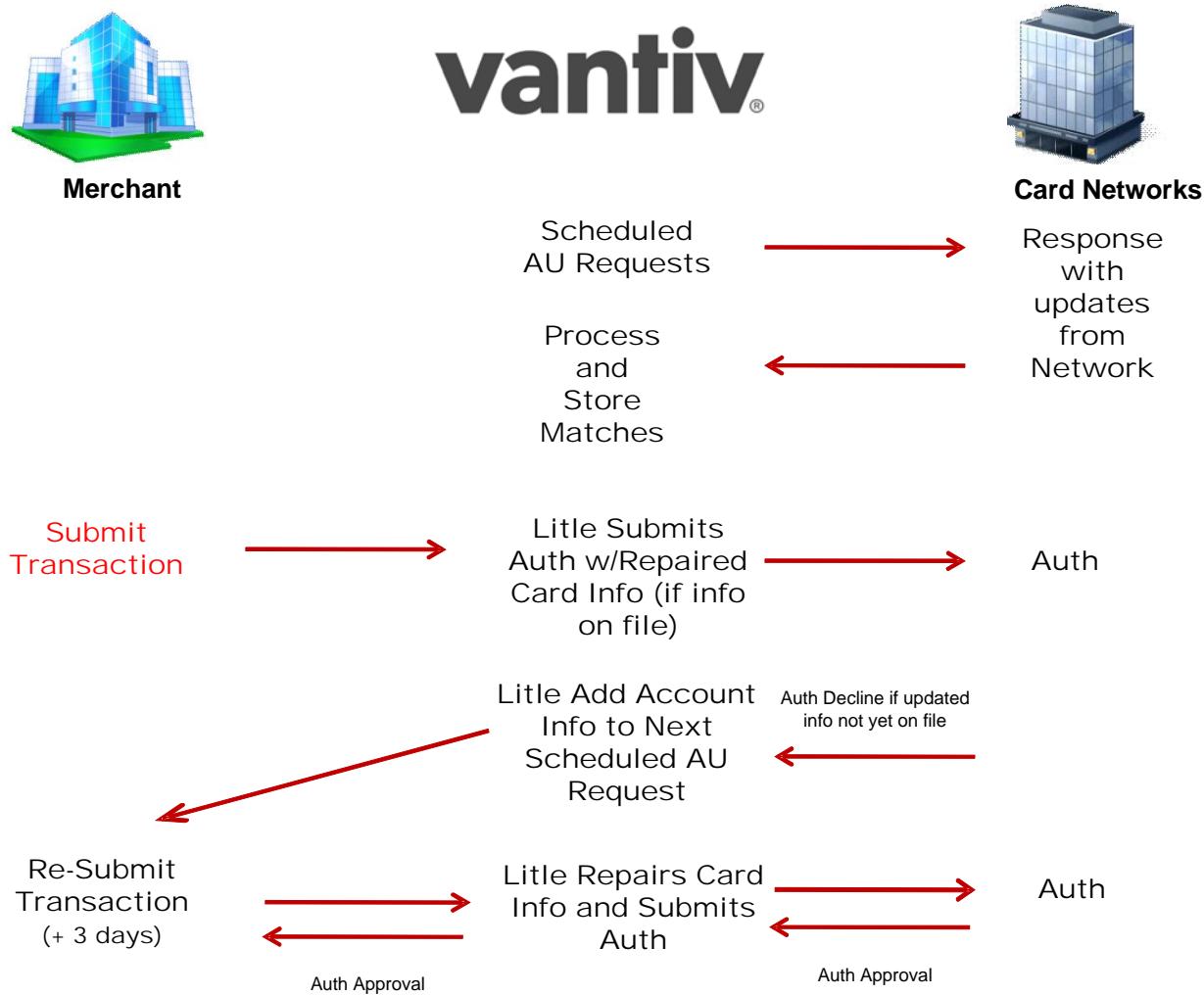
1.6.2 Account Updater Service

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 Account Updater service, the standard method for merchants to obtain updated account information was to submit a Batch containing existing card information, requesting that Vantiv 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 Vantiv 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.

The Account Updater service shifts the workload of obtaining and maintaining updated account information to Vantiv. Utilizing configurable scheduling algorithms, we initiate account update requests on your behalf and then stores the updated card information for use in future transactions. You simply submit billing transactions normally and, if necessary, Vantiv updates the transaction with the stored card information before submitting it to the networks for authorization. This fully managed service requires no code update on your systems.

FIGURE 1-3 Account Updater Overview



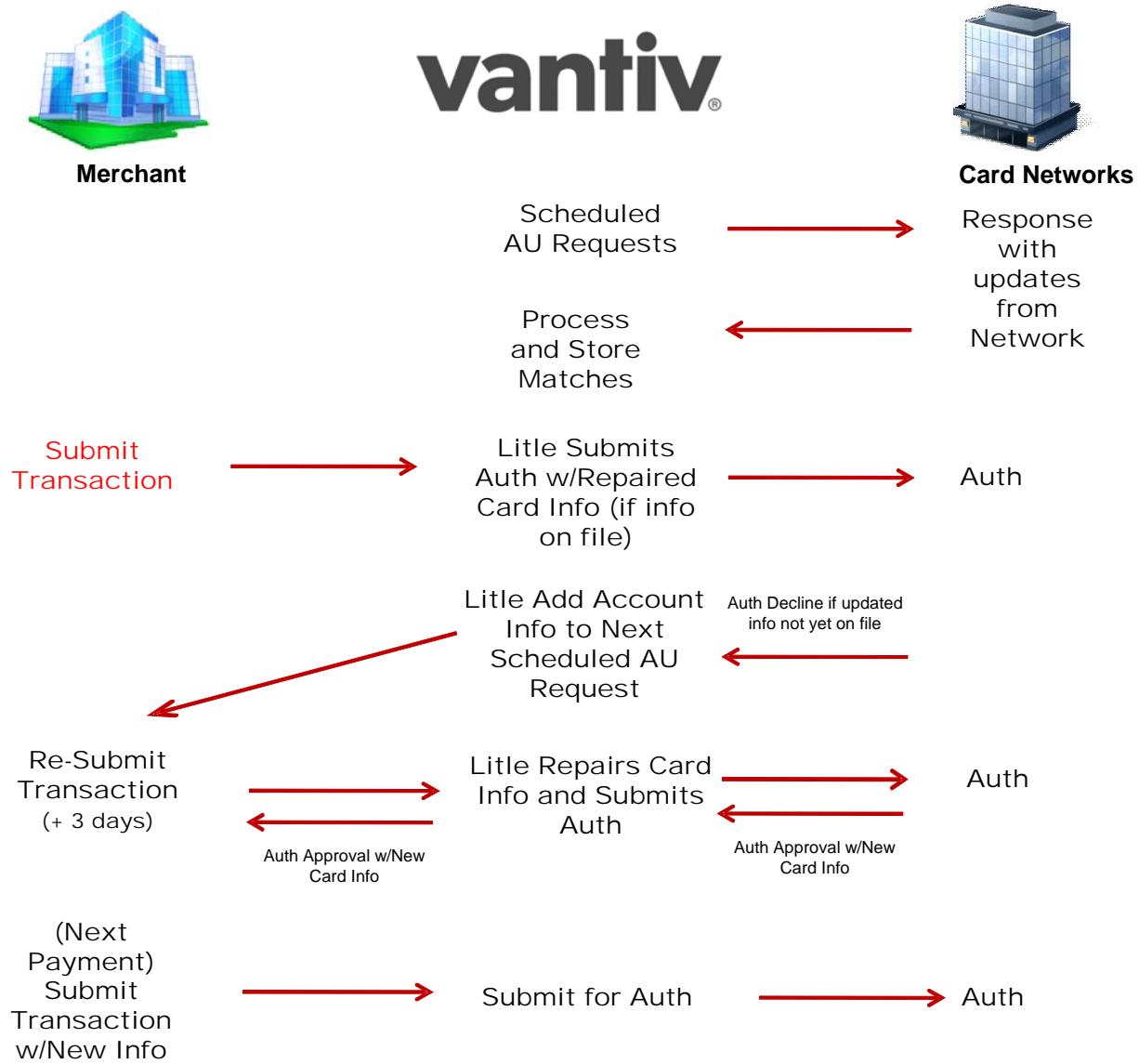
1.6.2.1 Match Back

If you decide you wish to have the updated card information returned to you, Vantiv offers the Match Back option. In this case, you can opt to receive updated information either in a Batch deposited to the merchant sFTP account, or in the XML transaction response messages. Once you update your systems, you can resubmit the failed transaction with the new card information. If,

after receiving an update, you submit a transaction with the old information, systems detect that you are using the old data and update the transaction for you prior to submitting it to the networks for authorization.

Please consult your Relationship Manager for additional information and configuration options.

FIGURE 1-4 Match Back Overview



1.6.2.2 Merchant Requirements

In order to use the 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 Vantiv acquired merchants)

Your Account Updater Welcome Kit includes the required application forms. If you have any questions about these forms, contact your 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**

1.6.2.3 Account Updater Features

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

- Vantiv initiates requests for updated account information to card networks based upon your billing cycle.
- Vantiv 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 LittleXML 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 Vault for merchants utilizing Vantiv's tokenization solution.
- Return of Extended Response Codes in the LittleXML response messages.

1.7 Recurring Engine

The Recurring Engine is a managed service that relieves the burden of developing an in-house billing solution for merchant engaged in installment or recurring transactions. This powerful, but flexible service allows you to create virtually any payment Plan required by your business model, whether it is part of a predetermined campaign or a marketing test, and then apply the Plan to customers as part of the standard Authorization or Sale transaction.

The Recurring Engine provides the following benefits:

- **Reduced Infrastructure Costs** - since you do not need to program your own solution, you save the up-front development cost, as well as ongoing maintenance expenses.
- **Reduced Labor** - once you create a Subscription in the Recurring Engine via a standard Authorization or Sale transaction, no further action is required for the life of the Subscription.
- **Integration with other Vantiv Value Added Services** - if you include the Recovery Services (Account Updater and Recycling Engine) as part of your implementation, you eliminate any concerns (and reduce expenses) associated with issues such as account number changes and recycling of declined payments.
- **Flexible Plans** - You can define the billing interval (weekly, monthly, quarterly, etc.), number of payments (including open ended schedules), amount of payments, as well as trial periods within a Plan. To add flexibility you can override several of these settings and set a specific start date at the individual Subscription level.
- **Flexible Creation of Discounts** - if you wish to offer a discount to selected customers, simply include the information at the time of the Subscription creation, or add it anytime afterward by updating the Subscription.
- **Flexible Creation of Add Ons** - similar to a discount, you can apply changes for additional services at the time of the Subscription, or anytime afterward.
- **Integrated Reporting** - in addition to the normal revenue reconciliation information available in the iQ Reporting and Analytics platform, there are a number of recurring specific reports that allow you to better analyze your revenue stream associated with recurring payment plans and strategies.

1.7.1 Payment Plans

The first step in setting up recurring billing on the Vantiv eCommerce platform is to establish one or more payment Plans. To establish a payment Plan you use a Create Plan transaction type, which allows you to define the payment interval, the number of payments, and the amount. For example, you could easily define any number of Plans to fulfill your business needs.

For example, suppose you are a SaaS company that sells your product under 1, 2, or 3 year deals, with either monthly or quarterly payment schedules and reduced rates for longer deals. You could easily set-up six Plans as shown in the table below.

TABLE 1-4 Example Plans

Plan Code	Payment Interval	Amount per Payment	# of Payments	Total Subscription
1_Year_Monthly	Monthly	\$50.00	12	\$600.00
1_Year_Quarterly	Quarterly	\$150.00	4	\$600.00
2_Year_Monthly	Monthly	\$46.66	24	\$1119.84
2_Year_Quarterly	Quarterly	\$140.00	8	\$1120.00
3_Year_Monthly	Monthly	\$41.66	36	\$1499.76
3_Year_Quarterly	Quarterly	\$125.00	12	\$1500.00

As part of the Plan, you can also specify trial period. You want to have longer trials for longer Plans, so for either 1-year Plan, there is a 1 week trial, for either 2-year Plan there is a 2 week trial, and for the 3-year Plans, a 1 month trial. Below is a LittleXML example transaction to create the 3_Year_Monthly Plan.

Example: 3-Year Monthly Plan

```
<createPlan>
  <planCode>3_Year_Monthly</planCode>
  <name>3Year_Monthly</name>
  <description>3 Year, monthly Payments, 1 month trial</description>
  <intervalType>MONTHLY</intervalType>
  <amount>4166</amount>
  <numberOfPayments>36</numberOfPayments>
  <trialNumberOfIntervals>1</trialNumberOfIntervals>
  <trialIntervalType>MONTH</trialIntervalType>
  <active>true</active>
</createPlan>
```

1.7.2 Subscriptions

Subscriptions marry a customer order to a particular payment Plan and initiate the Recurring Engine to manage your future billing. You create a Subscription using either an Authorization or a Sale transaction. In the Auth/Sale you simply include a `<recurringRequest>` element to initialize the Subscription using a named Plan and set the start date for the first recurring bill. If you do not include a start date, the Recurring Engine uses the current date for the first payment.

If the recurring bill had an associated set-up or one-time fee use a Sale transaction. The amount of the Sale transaction would represent that fee, whereas the amount of future recurring payments are defined in the Plan. If you use an Authorization to create the Subscription, the transaction would normally be a \$0 Auth (or small amount followed by a reversal) and would include the billing information for Address Verification.

As part of the Subscription creation, you can also override both the number of payments and the amount, as well as include Add Ons and Discounts (discussed in the next section). The overrides give you a granular control to modify a standard payments Plan for a particular consumer without creating additional Plans. For example, if you offered a 1-year Plan with monthly payments as shown in the previous section, you could allow a consumer to complete their payments in 10 months instead of a year. In this case you would override the number of payments defined in the Plan (12) with 10 payments, while increasing the amount of each payment from \$50 to \$60.

Example: Subscription with Overrides

```
<authorization id="834262" reportGroup="ABC Division" customerId="038945">
  <orderId>65347567</orderId>
  <amount>0</amount>
  <orderSource>ecommerce</orderSource>
  <billToAddress>
    .
    .
    .
  </billToAddress>
  <card>
    .
    .
    .
  </card>
  <recurringRequest>
    <subscription>
      <planCode>1_Year_Monthly</planCode>
      <numberOfPayments>10</numberOfPayments>
      <startDate>2016-09-21</startDate>
      <amount>6000</amount>
    </subscription>
  </recurringRequest>
</authorization>
```

1.7.2.1 Add Ons and Discounts

Occasionally, you might wish to modify a Subscription with either a Discount or an Add On without creating a new Plan that has limited use. A Discount reduces the recurring amount for one or more payments, while an Add On increases the payments in return for an added service or item. You can apply either of these payment modifications at the time you initialize the Subscription or anytime afterward by updating the Subscription. In both cases you define the start date, end date, and amount of the Discount/Add On.

For example, suppose as part of your standard offering, your customers received 2GB of cloud-based storage. You also offer additional storage in 2GB blocks for \$10 each. One of your customers wants an additional 4GB of storage for the 8 months remaining on his contract and you are discounting the first month at 50%, or \$10. The example below show the updateSubscription transaction with the Add On and Discount.

Example: Update Subscription with Discount and Add On

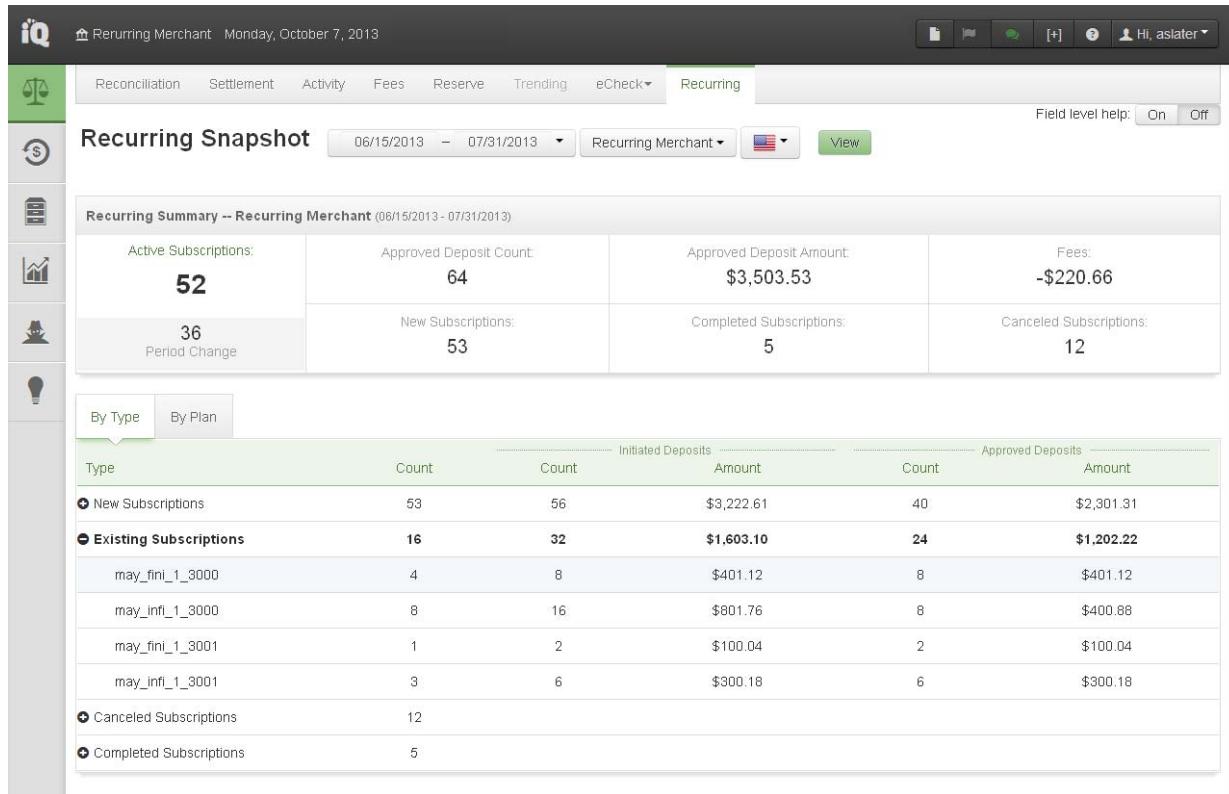
```
<updateSubscription>
  <subscriptionId>1234</subscriptionId>
  <createDiscount>
    <discountCode>4GBExtraDeal</discountCode>
    <name>Half-Off 1st Payment 4GB Extra</name>
    <amount>1000<amount>
    <startDate>2016-09-15</startDate>
    <endDate>2016-10-14</endDate>
  </createDiscount>
  <createAddOn>
    <addOnCode>4GB_Extra</addOnCode>
    <name>Four_GB_Extra</name>
    <amount>2000<amount>
    <startDate>2016-09-15</startDate>
    <endDate>2016-04-15</endDate>
  </createAddOn>
</updateSubscription>
```

1.7.3 Recurring Reports

In addition to recurring transactions appearing in the normal Vantiv reports (i.e., Payment Detail, Reconciliation report, etc.), there are currently two reports associated specifically with the Recurring Engine. The first report is a daily Recurring report available via sFTP. This report is in Batch format and contains LittleXML saleResponse messages for all recurring transactions run that day, including all approvals and declines (see the example, [Batch Sales Response with Recurring Info](#) on page 323).

The second report is the Recurring Snapshot report (see below). This report provides insights into the performance of each of your Plan offerings over a selected time period. The report includes metrics on Subscriptions added, completed, and cancelled, as well as information on any associated fees and amounts successfully deposited.

FIGURE 1-5 Recurring Snapshot Report



1.7.4 Transaction Types and Uses

The table below provides information about the various Recurring Engine transaction types and their uses.

TABLE 1-5 Recurring Engine Transaction Types

Use Case/Intent	Parent Element	Description/Uses
Create Plan	createPlan	Used to create new Plans.

TABLE 1-5 Recurring Engine Transaction Types

Use Case/Intent	Parent Element	Description/Uses
Update Plan	updatePlan	Used to toggle a Plan between an active and inactive state. You can not associate an inactive Plan with a Subscription. Toggling a Plan to an inactive state does not affect existing Subscriptions associated with the Plan.
Create Subscription	<authorization> or <sale> <recurringRequest> <subscription>	Used to initiate a Subscription with an associated Plan. The Subscription can include <code>amount</code> and <code>numberOfPayment</code> overrides to the Plan. Also, the Subscription can include <code>createAddOn</code> and <code>createDiscount</code> children.
Update Subscription	updateSubscription	Used to modify Subscription information including: changing the Plan, changing the billing name/address, and changing the method of payment. You can also use this transaction type to create/delete/update Add Ons and Discounts.
Cancel Subscription	cancelSubscription	Used to cancel an existing Subscription.
Create Add On	<authorization> or <sale> <recurringRequest> <subscription> <createAddOn> Or <updateSubscription> <createAddOn>	Used to create an Add On charge associated with the Subscription.
Update Add On	<updateSubscription> <updateAddOn>	Used to modify one or more of the parameters associated with the Add On.
Delete Add On	<updateSubscription> <deleteAddOn>	Used to delete an Add On charge from the associated Subscription.

TABLE 1-5 Recurring Engine Transaction Types

Use Case/Intent	Parent Element	Description/Uses
Create Discount	<authorization> or <sale> <recurringRequest> <subscription> <createDiscount> Or <updateSubscription> <createDiscount>	Used to create a Discount charge associated with the Subscription.
Update Discount	<updateSubscription> <updateDiscount>	Used to modify one or more of the parameters associated with the Discount.
Delete Discount	<updateSubscription> <deleteDiscount>	Used to delete a Discount from the associated Subscription.

1.8 Customer Insight Features

The Customer Insight 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, Vantiv offers four features in the Customer Insight family of features: Prepaid Indicator, Affluence Indicator, Issuer Country Indicator, Card Type Indicator.

1.8.1 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.

1.8.2 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.

Vantiv analysis of payments data indicates that consumers using these card 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 LittleXML 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.

1.8.3 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 successfullness 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 International Card Filtering feature to limit your exposure to international fraud originating in particular geographic locations.

1.8.4 Cardholder Type Indicator

The Card Holder Type indicator is an additional data point Vantiv can provide as part of the Customer Insight family of features. This indicator returns an element specifying 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.

1.8.5 Flow Control for the Insights Feature

Vantiv provides flow control functionality that allows you to limit which transactions the system examines and returns indicators. Currently there are three flow control options: by Presenter, by Report Group, and by Order Source. The **by Report Group** option allows you to limit the transactions considered for indicators to only those associated with a particular Report Group. The **by Presenter** option allows you to limit the transactions considered for indicators to only those from one or more presenters. The **by Order Source** option allows you to include or exclude transactions with a particular Order Source tag (`orderSource` element in the XML request). You can combine these Flow Controls. For example, you could establish a flow control such that the system would consider all transaction from Presenter A except those with an Order Source of recurring.

1.9 Fraud Toolkit

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 multiple reasons you may wish to decline a sale on a particular card at a particular time. In many cases there are indicators that the transaction could be or likely is fraudulent. Acting to stop these transactions at submission prevents loss, as well as reducing the number fraud related chargebacks in the future. Vantiv offers a robust fraud solution, the Fraud Toolkit, to assist you in reducing the number of possibly fraudulent transactions inflicted upon you by bad actors.

The Fraud Toolkit has three tiers or levels of implementation, each providing more rigorous examination of transaction properties and data points, as well as valuable information and guidance. The table below provide an overview of the tool provided at each level. The items highlighted in blue require the inclusion of a small snippet of code on your checkout page.

TABLE 1-6 Fraud Toolkit Implementation Levels

Filter/Feature	Essential	Extended	Premium
AVS Filter	X	X	X
CVV No-Match Filter	X	X	X
International BIN Filter	X	X	X
Prepaid Non-Reloadable Filter	X	X	X
Prior Chargeback Filter	X	X	X
Prior Fraud Advice Filter	X	X	X
Card Velocity Filter	X	X	X
Email Velocity Filter	X	X	X
Phone Velocity Filter	X	X	X
IP Velocity Filter	X	X	X
Device Velocity Filter	X	X	X
IP Address, Geolocation, and Proxy Detection		X	X
Merchant Customizable Rules Template		X	X
ThreatMetrix Cybercrime Dashboard		X	X
(Asynch) Transaction Review Queues		X	X
Rule Management and Portal Training		X	X
Standalone Transaction API Access		X	X
Cybercrime Industry Report (Quarterly)		X	X

TABLE 1-6 Fraud Toolkit Implementation Levels

Filter/Feature	Essential	Extended	Premium
Access to Fraud Consultant			X

1.9.1 Essential Tier

The Essential tier includes a suite of eleven Fraud Filters that you can apply individually or in combination. Nine of the eleven filters, based on card/submitted data, potentially require no additional integration on your part, assuming you already submit the necessary information. The remaining two filters require you to add a small snippet of code on your checkout page.

NOTE: Technically, you can make use of the IP Velocity filter without integrating the code snippet on your checkout page. Instead you can simply include the originating IP Address that you detect in your transaction by submitting it as a `customAttribute`. Please note that this method will likely be less effective than making use of the ThreatMatrix functionality, which includes IP piercing to determine the true IP of the consumer's device.

1.9.1.1 Prepaid Card Filtering

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 Vantiv filter these transactions automatically when you send the Authorization transaction.

If you elect to use the 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. Upon a decline, the system returns a Response Reason Code of **309 - Restricted Card - Prepaid Card Filtering Service**. 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**, allowing you to accept any prepaid card for these transactions.

The second method of implementing the Prepaid Card Filtering Service is per transaction. To enable the filter on a particular transaction, set the `<prepaid>` element to a value of **true**. This method is 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 prepaid cards.

NOTE:

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

1.9.1.2 International BIN Filtering

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

If you elect to use this feature, when you submit an Authorization/Sale transaction, the 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. Upon a decline, the system returns a Response Reason Code of **312 - Restricted Card - International Card Filtering 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.

1.9.1.3 Prior Chargeback Filtering

If you elect to use the Chargeback Filter Service, there are two configuration options. You can elect to filter all transactions using a card for which you received a chargeback, or you can elect to filter only the subset of transactions for which you received a fraud related chargeback (determine by the associated chargeback reason code). In both cases, the system checks your historical data to see if you received an applicable chargeback from the same account within the last 90 days. Upon a decline, the system returns a Response Reason Code of **308 - Restricted Card - Chargeback**.

1.9.1.4 Security Code No-Match Filter

The Card brands added the 3- or 4-digit security code to act as a verification that the person ordering your product in a card-not-present environment has physical possession of the card. While this validation can be a useful anti-fraud tool, typically, many issuing banks do not decline the transaction based upon a failure to match the security code. Declining the transaction is left to the discretion of the merchant.

NOTE:	<p>Since American Express declines the transaction when the security code does not match, the Security Code No-Match filter does not apply to American Express transactions. Transactions declined by American Express for a failure to match the security code use the Response Reason Code of 352 - Decline CVV2/CID Fail.</p> <p>Similarly, if Visa, MasterCard, or Discover decline a transaction based upon the security code results, Vantiv does not apply the filter and the transaction response contains the 352 Reason Code.</p>
-------	---

If you elect to use the Security Code No-Match Filter Service, the system takes action only if the issuer approves the submitted authorization/sale transaction, but includes a no-match code for the CVV2/CVC2/CID card validation check. In this case, the Vantiv declines the transaction with a Response Reason Code of **358 - Restricted by Little due to security code mismatch**. The system also issues an Auth Reversal transaction on your behalf to remove the funds hold on the account.

1.9.1.5 Card Velocity Filtering

Often, when a person attempts to use a stolen credit card successfully, they will follow the initial purchase with a number of additional purchases within a short period of time. If you elect to use the Fraud Velocity Filter, the system filters the transaction based upon the number of previously approved Auth/Sale transactions plus the number of Auth/Sale transactions declined by another filter, for the same account within a designated time period. Both the total number of transactions and the time period are configured in the Vantiv Merchant Profile.

Upon a decline, the system returns a Response Reason Code of **315 - Restricted Card - Auth Fraud Velocity Filtering Service**.

1.9.1.6 Prior Fraud Advice Filtering

Vantiv maintains a database of Fraud Advice information received from the Visa and MasterCard networks for transactions you processed in the last 200 days. If you use the Prior Fraud Advice Filter, the system compares the account information from the new transaction against the database of accounts with prior Fraud Advice and filters the transaction if there is a match.

Upon a decline, the system returns a Response Reason Code of **318 - Restricted Card - Auth Fraud Advice Filtering Service**.

1.9.1.7 AVS Filter

One of the fraud prevention tools provided by all card networks is an Address Verification System. By submitting the customer's address information in the `billToAddress` section of the LittleXML message, you can verify that the address/zip code supplied by the consumer matches the issuer's records. The card networks, however, do not decline transactions based upon the

failure to match the address or zip code. Using the AVS Filter, you can filter potentially fraudulent transactions based upon failure to match any of the following:

- the address
- the zip/postal code
- the address + zip/postal code (ANDed)
- the address or zip/postal code (ORed).

Upon a decline, the system returns a Response Reason Code of **319 - Restricted Card - Fraud AVS Filtering Service**.

1.9.1.8 Email Velocity Filter

Often, card testers or other bad actors submit a number of transaction using multiple cards, but with a common email address. The only requirement to make use of this filter is that you collect and include the consumer's email address with each transaction. We communicate the email address to our fraud partner, ThreatMetrix, who tracks and analyzes the information. If the filter detects the same email used in the configured number of transactions within the configured period of time, the system declines new transactions (using the same email) on your behalf and returns Response Code **550 - Restricted Device or IP - ThreatMetrix Fraud Score Below Threshold**.

1.9.1.9 Phone Velocity Filter

Similar to email, card testers or other bad actors often submit a number of transaction using multiple cards, but with a common phone number. The only requirement to make use of this filter is that you collect and include the consumer's phone number with each transaction. We communicate the phone number to our fraud partner, ThreatMetrix, who tracks and analyzes the information. If the filter detects the same phone number used in the configured number of transactions within the configured period of time, the system declines new transactions (using the same email) on your behalf and returns Response Code **550 - Restricted Device or IP - ThreatMetrix Fraud Score Below Threshold**.

1.9.1.10 IP Velocity Filter

The IP Velocity filter is one of the two filter in the Essential tier that requires (see note below) the addition of a code snippet to your checkout page. This snippet, which you also need to implement for the higher tiers of Fraud Toolkit, allows our partner, ThreatMetrix, to perform IP interrogation/piercing to determine the true IP Address of the device originating the order. As with the other velocity filters, if the filter detects the same IP Address used in the configured number of transactions within the configured period of time, the system declines new transactions from the same IP Address on your behalf and returns Response Code **550 - Restricted Device or IP - ThreatMetrix Fraud Score Below Threshold**.

NOTE: Technically, you can make use of the IP Velocity filter without integrating the code snippet on your checkout page. Instead you can simply include the originating IP Address that you detect in your transaction. Please note that this method will likely be less effective than making use of the ThreatMetrix functionality, which includes IP piercing to determine the true IP of the consumer's device.

1.9.1.11 Device Velocity Filter

The Device Velocity filter is the second Essential tier filter in the that requires the addition of a code snippet to your checkout page. In this case, the snippet allows ThreatMetrix to construct a device fingerprint of the system originating the order. As with the other velocity filters, if the filter detects the same device used in the configured number of transactions within the configured period of time, the system declines new transactions from the same device on your behalf and returns Response Code **550 - Restricted Device or IP - ThreatMetrix Fraud Score Below Threshold**.

1.9.1.12 Application of Filters - Filtering Rules

NOTE: Filter Rules are defined as part of your Merchant Profile. Please consult with your Relationship Manager and/or your Implementation Consultant concerning the provisioning of Filter Rules.

While you can have all submitted transactions flow through the Fraud toolkit, you likely want to exercise a finer control over the application of the filters based upon a particular product, service or other criteria. The system provides you the flexibility of restricting which transactions are submitted to the filtering service and which filters the system applies to which groups. This is accomplished by defining Filtering Rules.

For each Filtering Rule you first define a subgroup of transactions by selecting one of the following Flow Selectors: Report Group, Billing Descriptor, orderSource, or MID (for PayFacs, flow control by MID or order source only). Only one selector can be applied per rule. After selecting a particular Flow Selector, you then select which filters to have applied to that subset of transactions. You can define the Filter Rules so that filters are ORed (transaction filtered when any one of the filters conditions met), or ANDed (transaction filtered when multiple filter conditions met). [Table 1-7](#) defines five rules that a merchant might define.

TABLE 1-7 Example - Fraud Filtering Service Rules

Filter	Flow Selector	Filters
1	Report Group = "XYZ"	Prepaid

TABLE 1-7 Example - Fraud Filtering Service Rules

Filter	Flow Selector	Filters
2	Report Group = "XYZ"	International
3	orderSource = "recurring"	Prepaid + Prior Chargeback
4	orderSource = "ecommerce"	Card Velocity + Security Code No-match
5	Billing Descriptor = "GoldMember"	Prepaid + International

Table 1-7 defines five Filter Rules that a merchant might use. These rules would be applied as follows:

- Filters 1 and 2 apply to the subset of transactions that are members of Report Group XYZ and use the Prepaid and International Filters. Since the Filter Rules are defined separately, the rules are ORed. So, if a transaction uses either a Prepaid card or a card of International origin, the filtering occurs.
- Filter 3 applies to the subset of transactions that have an orderSource value set to recurring. Filtering occurs only if both the criteria for the Prepaid Filter AND the Prior Chargeback Filter are met.
- Filter 4 applies to the subset of transactions that have an orderSource value set to ecommerce. Filtering occurs only if both the criteria for the Card Velocity Filter AND the Security Code No-Match Filter are met.
- Filter 5 applies to the subset of transactions that have an Billing Descriptor value set to GoldMember. Filtering occurs only if both the criteria for the Prepaid Filter AND the International Filter are met.

1.9.2 Extended Tier

The Extended Tier include all of the Essential Tier filters, but offers an additional levels of fraud detection made available through Vantiv's partnership with ThreatMetrix. The addition of the same code snippet used for the IP and Device Velocity filters to your checkout page allows ThreatMetrix to gather additional data points, such as the consumer's device, proxy use, and location. Unlike the filters in the Essential Tier, which are basic accept/decline filters, the Extended Tier takes the data and compares the information to a rule list. Vantiv supplies an initial, Best Practices rules list designed for your business type (i.e., Retail, Digital, Non-profit, etc.), which you can modify and refine for you particular business model. Each rule, when triggered, add or subtract a preset value from the transaction score. If the score fall below a set threshold, the system declines the transaction, unless you prefer to make the final decision yourself. In either case, Vantiv returns the score and a list of triggered rules in the transaction response message.

In addition to the ThreatMetrix rules engine, you get access to the ThreatMetrix Portal allowing you to customize your rules list and scoring values. This level also allows you to white list/black list items, such as email addresses and phone numbers. Other items included in this tier are:

- ThreatMetrix Cybercrime Dashboard

- Asynchronous Transaction Review Queues
- Monthly Rules and Portal training
- API Access to Standalone Fraudcheck transaction.

1.9.3 Premium Tier

The Premium Tier provides all of the tools from the Essential and Extended Tiers, and most importantly, access to the Vantiv eComm Fraud Consulting service. With the service, the Fraud Consultant assigned to you helps analyze your transactional data, recommend rule changes to fine-tune your results, and advises you on fraud detection strategy.

1.9.4 Modifications to Your Web Page

For ThreatMetrix to gather information for analysis, you must add certain profiling tags (see example below) to selected pages served by your web application. These tags allow ThreatMetrix to collect information by loading objects used for detection into the consumer's browser. These tags are invisible to the consumer and add only a fraction of a second to your page's rendering time. Once loaded, these objects require only 3-5 seconds to gather profiling information from the consumer device.

Place the tags as early as possible on the page, inside the `<body></body>` tags of the page HTML.

Example: ThreatMetrix Profiling Tags

NOTE: Replace `UNIQUE_SESSION_ID` with a uniquely generated handle that includes the Vantiv supplied prefix.
The value for `ORG-ID` is a Vantiv supplied value.
The `pageid` tag is not used at this time. The value for `PAGE-ID` defaults to 1.
For production, replace `h.online-metrix.net` with a local URL and configure your web server to redirect to `h.online-metrix.net`.

```
<!-Begin ThreatMetrix profiling tags below -->
<script type="text/javascript"
src="https://h.online-metrix.net/fp/tags.js?org_id=ORG_ID&session_id=UNIQUE_SESS
ION_ID&pageid=PAGE_ID"></script>

<noscript>
<iframe style="width: 100px; height: 100px; border: 0; position: absolute; top:
-5000px;" src="https://h.online-metrix.net/tags?org_id=ORG_ID&session_id=UNIQUE_SESSION_ID
&pageid=PAGE_ID"></iframe>
```

```
</noscript>  
!-- End profiling tags -->
```

1.9.4.1 LittleXML Transactions

To subject a transaction to the advanced fraud checks performed by ThreatMetrix and retrieve the results, you simply submit the <threatMetrixSessionId> element as part of your LittleXML Authorization (or Sale) transaction. This session Id is the same unique value you assigned and sent to ThreatMetrix when your web page called the application (designated as UNIQUE_SESSION_ID in the ThreatMetrix Profiling Tags example). When we receive an Authorization/Sale that includes the <threatMetrixSessionId>, our system automatically queries the ThreatMetrix platform for the associated results. The LittleXML response message includes the <advancedFraudResults> element containing the score and status and information about any triggered rules. The following two examples show a standard Authorization transaction, including a <threatMetrixSessionId> and a **pass** response.

Example: Authorization including <threatMetrixSessionId> Element

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"  
merchantId="81601">  
  <authentication>  
    <user>User Name</user>  
    <password>password</password>  
  </authentication>  
  <authorization id="002" reportGroup="001601">  
    <orderId>10102013_sessionId_app</orderId>  
    <amount>1002</amount>  
    <orderSource>ecommerce</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>nobody@little.com</email>  
    </billToAddress>  
    <card>  
      <type>MC</type>  
      <number>5405102001000003</number>  
      <expDate>1115</expDate>  
    </card>  
  </advancedFraudChecks>
```

```

        <threatMetrixSessionId>ASDFG-AXXXXAB999</threatMetrixSessionId>
    </advancedFraudChecks>
</authorization>
</littleOnlineRequest>

```

Example: Authorization Response including <advancedFraudResults> Element

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">

    <authorizationResponse id="002" reportGroup="001601">
        <littleTxnId>82823534116454639</littleTxnId>
        <orderId>10102013_sessionId_app</orderId>
        <response>000</response>
        <responseTime>2017-03-08T21:36:50</responseTime>
        <postDate>2017-03-08</postDate>
        <message>Approved</message>
        <authCode>000003</authCode>
        <fraudResult>
            <avsResult>00</avsResult>
            <advancedFraudResults>
                <deviceReviewStatus>pass</deviceReviewStatus>
                <deviceReputationScore>50</deviceReputationScore>
                <triggeredRule>FlashImagesCookiesDisabled</triggeredRule>
            </advancedFraudResults>
        </fraudResult>
    </authorizationResponse>
</littleOnlineResponse>

```

NOTE: The other possible values for the <deviceReviewStatus> element are *fail*, *review*, *unavailable*, and *invalid_session*.

The <deviceReputationScore> value can range from -100 to 100. The resulting *pass*, *fail*, or *review* value depends upon your profile settings.

The <triggeredRule> element can occur multiple times, once for each rule triggered.

1.9.4.2 Information Only Option

If you wish to retain full control of the decision to accept or decline transactions, Vantiv offers the option of using the Advanced Fraud Tools in an Information Only mode. In this configuration, you receive the same information in the response as you would with the full implementation; however, Vantiv will not automatically decline transactions with a failing score.

If the authorization is declined by the network, you can choose to recycle the transaction or do nothing. If an authorization with a failing score receives approval from the network, it would be up to you to reverse the authorization should you decide not to proceed with the transaction. This is similar to the case of an approved transaction that has a status of Review, but you decide not to proceed. Issuing an authorization reversal allows you to avoid any misuse of Auth fees otherwise imposed by the card networks.

1.9.4.3 Fraud Check Transactions

If you wish to retrieve the Advanced Fraud results without introducing a Authorization or Sale transactions, use a Fraud Check transaction (see [Fraud Check Transaction](#) on page 288). Fraud Check transactions are only supported as Online transactions.

1.10 Tokenization Feature

Tokenization is the process by which a reference number, referred to as a token, replaces a credit card number or eCheck account number. Unlike the card or account number, you can store the token on your system without concern of a security breach exposing critical customer information. Instead, Vantiv 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 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 in the event of a breach. You can reduce the requirements further, as well as the possibility of exposure from a breach, through the use of the Vantiv eProtect™. By sending the card information from your checkout page directly to our systems you eliminate one more facet of handling the card information.

NOTE: Vantiv 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:

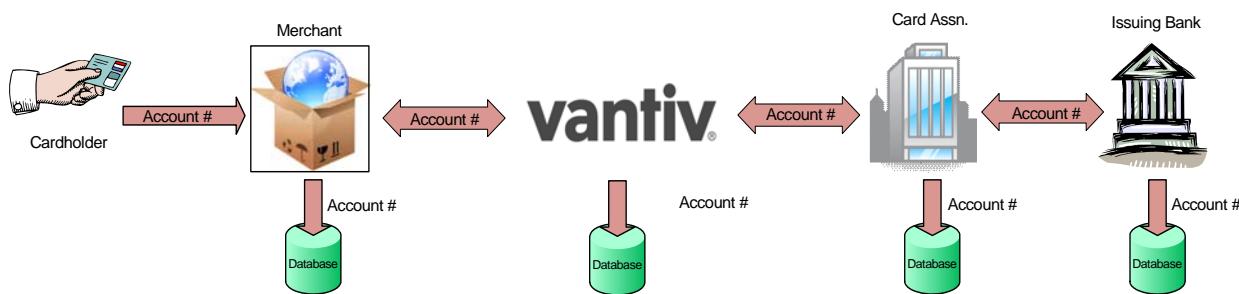
- How Tokenization Works
- Token Formats
- Obtaining Tokens
- Supported Token Transactions

NOTE: Please refer to the *Vantiv eProtect Integration Guide* for additional information about the use of and integration to the Vantiv eProtect value added service.

1.10.1 How Tokenization Works

In a non-tokenized environment, multiple parties handle and store customer data, including the card/eCheck account number, 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 shown in [Figure 1-6](#) 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 data transmission occurs 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 a malicious third party could compromise the information.

FIGURE 1-6 Card Information Flow in Non-Token Environment



In a tokenized environment transmission of customer data ideally occurs a single time and the merchant never stores it locally, as shown in [Figure 1-7](#) for card data. Once account number registration occurs, using either a `registerTokenRequest` or by submitting the account number (or low value token, when using eProtect) with any supported transaction, Vantiv returns a (high value) token. You store the token locally and use it for all future transactions concerning that account. Vantiv 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 Vantiv are different. The operation remains the same from a merchant standpoint.

FIGURE 1-7 Card Information Flow in Tokenized Environment

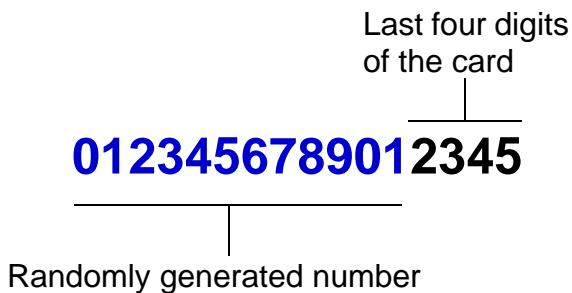
NOTE: The use of eProtect allow the account information to come directly to Vantiv, so the merchant handles the token only.

1.10.2 Token Formats

For credit cards, in an effort to minimize development requirements on the merchant side, Vantiv 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 1-8 Token Format - Card



For an eCheck token, since the account number length can vary widely, we 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.

1.10.3 Obtaining Tokens

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

There are three ways for you to obtain tokens for account numbers. First, you can submit an existing card number/eCheck account information (account number and routing number) using a Register Token request. When Vantiv receives this transaction type, we generate a token and return it to you via a Register Token response (see [Register Token Transactions](#) on page 307.)

Although you can use this method to tokenize an account number at any time, it is most useful when initially tokenizing your customer database. Vantiv recommends that you collect all distinct credit card numbers in your database and submit the information in one or more large Batch. 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, Vantiv 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 we convert a card number to a token for a particular merchant, subsequent submissions of the same card number return the same token.

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

1.10.3.1 Bulk Token Registration

If you are new to Vantiv, and have utilized tokens with a previous processor, Vantiv can perform a bulk token registration on all the card numbers that were vaulted with your previous processor. The following is an example of the process:

1. During your implementation with Vantiv, you contact your previous processor and request an encrypted mapping file containing the card and token numbers for your customers. A Implementation Consultant will work with you and your previous processor to facilitate the secure transfer of this file without impacting your PCI compliance. The file can be comma-delimited, tab-delimited, or any other common format.
2. Vantiv performs a bulk token registration of all of the card numbers contained in the file.
3. Vantiv returns a mapping file to your organization containing the old tokens and the new Vantiv-issued tokens, so that you can update your order processing system.

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

1.10.4 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, Vantiv 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, Vantiv 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, Vantiv 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, Vantiv 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 Vantiv 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, Vantiv 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, Vantiv 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, Vantiv 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, Vantiv 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, Vantiv automatically generates the token and returns it in the response.
- **Update Card Validation Number** - This is a special transaction type provided to allow the update of a CVV2/CVC2/CID code supplied at the time of the token registration. You should only use this transaction type if you had previously submitted the account number and security code in a `registerTokenRequest` transaction and now need to change the CVV2/CVC2/CID value.

1.10.5 Compliance with Visa Best Practices for Tokenization

As shown below, the 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 Vantiv solution would satisfy this item, as would protecting it per PCI DSS requirements.

TABLE 1-8 Visa Best Practices for Tokenization Compliance

Item #	Who	Domain	Best Practice	Complies?
1	Vantiv	Tokenization System	Network Segmentation	Yes
2	Vantiv	Tokenization System	Authentication	Yes
3	Vantiv	Tokenization System	Monitoring	Yes
4	Vantiv	Tokenization System	Token Distinguishability	Yes
5	Vantiv	Token Generation	Token Generation	Yes
6	Vantiv	Token Generation	Single- vs. Multi- use Tokens	Yes
7	Vantiv	Token Mapping	PAN Processing	Yes
8	Vantiv	Card Data Vault	PAN Encrypted in Storage	Yes
9	Vantiv	Card Data Vault	Covered by PCI DSS	Yes
10	Vantiv	Cryptographic Keys	Key Strength	Yes
11	Vantiv	Cryptographic Keys	Covered by PCI DSS	Yes
12	Merchant	Historical Data Management	Non-tokenized data protected	Merchant Implementation Decision

1.11 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 in the United States and no interchange fees.

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

NOTE: **Beginning in March, 2015, Vantiv also supports eCheck processing for our merchants doing business in Canada. All eCheck transaction types, except Verification and Prenotification transaction are supported. Please consult your Relationship Manager for additional information.**

1.11.1 Validation Feature

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

1.11.2 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: **Vantiv makes use of a third party service, Certegy Check Services Inc., for all verification operations.**
The verification service is not supported for Canadian eChecks.
The verification service is not available for PayFacs.

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.

1.11.2.1 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, Vantiv 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

1.11.3 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.

Vantiv 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 LittleXML response message to you also contains the updated information for your use in correcting your database.

1.11.4 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. Vantiv offers an optional service that allows you to preconfigure automatic redeposits of transactions returned for those reasons. You define the number of days from the initial return for Vantiv 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 iQ User Interface. Please refer to the *iQ Reporting and Analytics User 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 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 system generates the second redeposit on 14 February.

1.11.5 eCheck Prenotification

An eCheck Prenotification is a non-monetary transaction used to verify the account information supplied by the consumer is valid. These transactions are sent to the ACH network to help ensure subsequent entries are posted appropriately. Since this is a verification of account information, typically, you would submit a Prenotification transaction in advance of processing the order, during the customer set-up process. There are two types of Prenotification transaction types:

echeckPreNoteCredit and eCheckPreNoteSale. Per NACHA requirements, you must submit the Prenotification transaction that corresponds to the intended, subsequent transaction. For example, if you are planning to submit an echeckSale transaction and want to verify the account information, you should submit a echeckPreNoteSale.

The possible ACH network responses to a prenotification are as follows:

- **No response** - applies when the account is open and the account information is correct.
- **Notification of Change** - provides updated account information, including correct routing number, e.g. C02 Incorrect routing/transit number (visible in the SSR eCheck NOC report).
- **Return code** - provides account status, e.g. R02 Account is closed, R03 No account on file, R04 Invalid account number.

NOTE: **Prenotification transactions are only supported in LittleXML V9.1 or above and only in Batch submissions. Prenotification transactions are not supported for PayFacs or for Canadian eChecks.**

When you submit either of the Prenotification transaction types, the system sends an acknowledgment in the Batch response file. This response file does not provide the results of the prenotification check, but rather a verification that we received the transaction and it was properly formatted. You receive the results to the check in the SSR eCheck Notification of Change report. This report is run daily and you can expect to see the results for submitted prenotification checks by the second or third business day after the settlement day. The settlement day for a Prenotification transaction is defined as the next business day after submission to the ACH network. Remember, if the account you are attempting to verify is open and the account information is correct, the report will not contain an entry for that transaction. The report only contains NOCs (update information).

Per NACHA regulations, you can submit the eCheck Sale or eCheck Credit transaction on the third business day after the settlement day; however, there might still be a forthcoming NOC on that day. Due to the timing of the responses from NACHA, the generation of the reports, and the movement of the transactions, you should wait until the fourth day after the settlement day to submit the live transaction unless you receive a NOC earlier. This timing is illustrated in the example below.

1. You submit a Prenotification transaction on Monday prior to your cut-off time.
2. Our system forwards the transaction to NACHA on Tuesday. Note, this makes Wednesday the settlement day.
3. NACHA responds with a NOC/return, if any, by Thursday night.
4. On Friday, the information from NACHA is processed by our systems.
5. On Saturday morning, the SSR NOC report containing the information is available to you.
6. You can submit the eCheck Sale or eCheck Credit transaction before your cut-off time on Saturday night. This is the fourth day after settlement, the fifth day after submission.

1.12 SEPA Direct Debit

When building a robust international ecommerce offering, you should always consider regional, preferred payment methods. Enabling payment methods with which consumers are both familiar and comfortable may provide a significant bump in your conversion rates and sales. In Europe there are dozens of regionally popular alternative payment methods. SEPA (Single Euro Payments Area) Direct Debit is a Pan-European network facilitating direct debit transactions in 34 countries. Currently, SEPA Direct Debit accounts for approximately 10% of all eCommerce sales in Europe and leads all alternative payment methods in Germany, accounting for more than 35% of eCommerce spending.

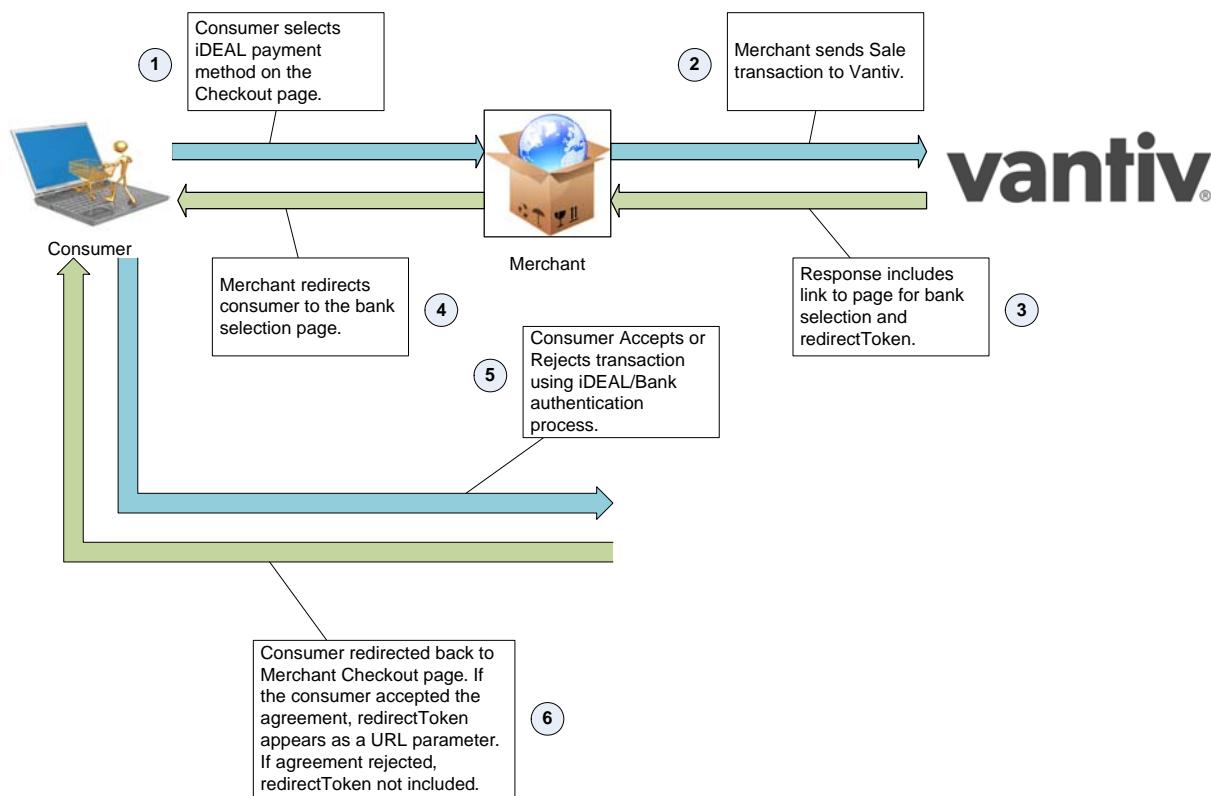
Although similar to eCheck transaction in the US, in that they are both direct debit methods of collecting funds, there are several differences in the process. Most of the difference concern the requirement for the presentation of a Mandate to the consumer. A Mandate is basically an agreement between the consumer and the merchant allowing the merchant to debit the consumer's account for the required funds. The agreement can specify either a one-time payment or recurring payments. The consumer must agree to the Mandate (Usually, by clicking a Confirm button on the Mandate page.) for the transaction to succeed.

The Vantiv eComm platform allows for two different Mandate scenarios. The first, and recommended method, involves the redirect from your site to a Vantiv provided Mandate page. Alternatively, you can provide your own Mandate page. The following sections discuss the purchase/order flow for each scenario.

1.12.1 SEPA Direct Debit - Vantiv Supplied Mandate

You cannot complete a SEPA Direct Debit transaction without an approved (by the consumer) Mandate. While you can produce your own Mandate page, Vantiv makes the implementation of SEPA Direct Debit easier for you by providing Mandates in various languages, for either single payments or recurring transactions.

Figure 1-9 provides an overview of the order flow, when using a Vantiv supplied Mandate for a one-time payment, or the first payment in a recurring stream. The steps provide additional information.

FIGURE 1-9 Order Flow for Vantiv Supplied Mandate

1. On your checkout page, the consumer selects SEPA Direct Debit as the payment method, enters their IBAN (International Banking Account Number), selects a preferred language for the Mandate (optional), and clicks the Submit button.

NOTE: **If you do not specify a preferred language, or you specify an unavailable language, the Mandate page appears in English.**

2. You create a Sale transaction and submit it to the Vantiv eComm platform, using `sepaDirectDebit` as the payment method and including its child elements shown below (`preferredLanguage` optional). In addition to the required child elements of `sepaDirectDebit` you must include the `name`, `email`, and `country` child elements of `billToAddress`.

Example: SEPA Direct Debit Sale Request Message - Vantiv Supplied Mandate

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema">
  <merchantId>SddMer1</merchantId>
  <authentication>
    <user>SDD</user>
    <password>password</password>
```

```

</authentication>
<sale id="SEPA_Sale" reportGroup="SddMer1" customerId="SEPA_Customer1">
  <orderId>sddSale</orderId>
  <amount>1001</amount>
  <orderSource>ecommerce</orderSource>
  <billToAddress>
    <name>Johann Schmidt</name>
    <addressLine1>10 Hoch Strasse</addressLine1>
    <city>Hanover</city>
    <zip>30159</zip>
    <country>DE</country>
    <email>jschmidt@phoenixprocessing.com</email>
    <phone>781-270-1111</phone>
  </billToAddress>
  <sepaDirectDebit>
    <mandateProvider>Vantiv</mandateProvider>
    <sequenceType>OneTime</sequenceType>
    <iban>DE79850503003100180568</iban>
    <preferredLanguage>DE</preferredLanguage>
  </sepaDirectDebit>
</sale>
</littleOnlineRequest>

```

3. The response message (example shown below) includes the redirectUrl element, which hosts the Mandate. The redirectToken returned in the response allows you to verify the consumer accepted the Mandate. If this transaction is the first of a recurring stream, you use the mandateReference element in subsequent payments to reference the agreed to (recurring) Mandate.

Example: SEPA Direct Debit Sale Response Message - Vantiv Supplied Mandate

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <saleResponse id="SEPA_Sale" reportGroup="SddMer1">
    <littleTxnId>82830949823203015</littleTxnId>
    <orderId>sddSale</orderId>
    <response>000</response>
    <responseTime>2017-01-21T17:07:27</responseTime>
    <postDate>2017-01-21</postDate>
    <message>Approved</message>
    <sepaDirectDebitResponse>
      <redirectUrl>http://MandateHostURL/DE</redirectUrl>
      <redirectToken>evr0kij413g8r85e32v7deov0c</redirectToken>
    </sepaDirectDebitResponse>
  </saleResponse>
</littleOnlineResponse>

```

```

<mandateReference>1ABCA43</mandateReference>
</sepaDirectDebitResponse>
</saleResponse>
</littleOnlineResponse>

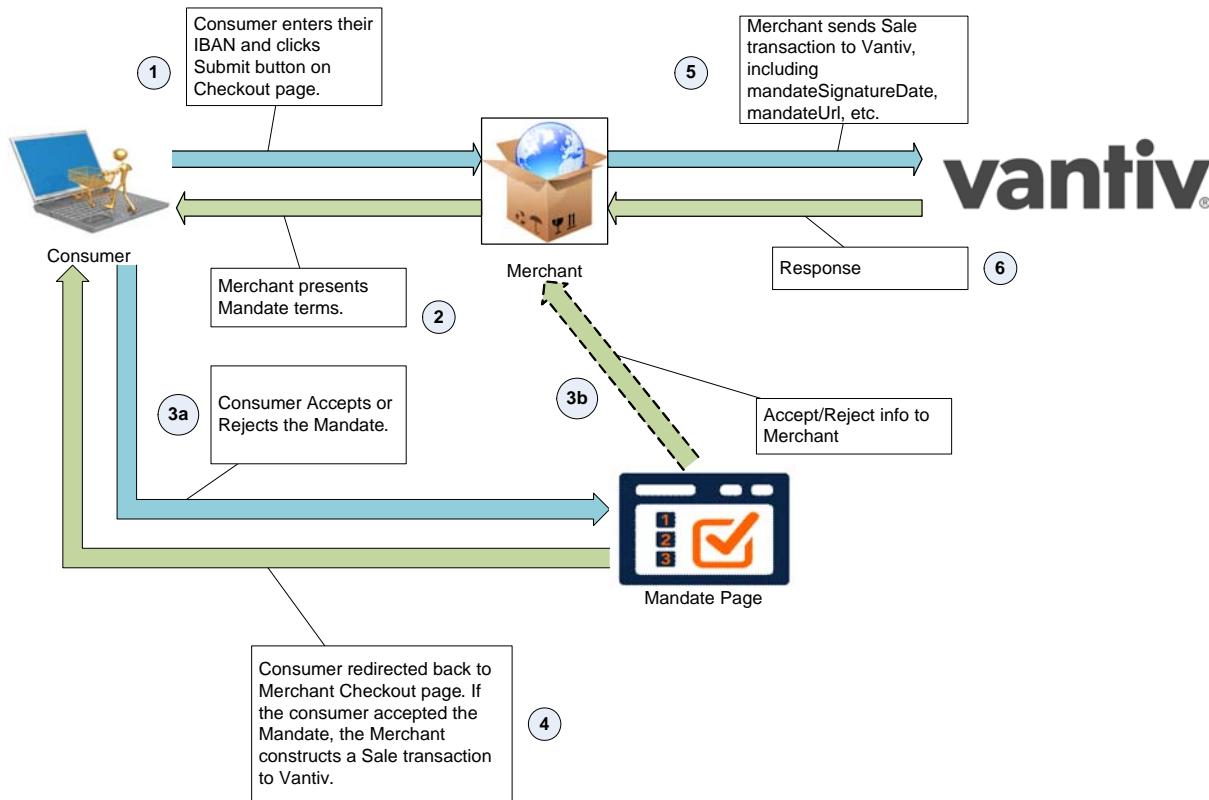
```

4. You redirect the consumer from your checkout page to the `redirectUrl`.
5. The consumer either accepts or rejects the Mandate.
6. After the consumer either accepts or declines the Mandate, they are redirected back to your checkout page, or alternate page upon decline. If they accepted the Mandate, the URL contains a parameter exposing the `redirectToken` value (`redirectToken=token_value`). By comparing this value to the value you received in the Sale response message you can verify that the consumer accepted the Mandate. If they declined the Mandate, the URL does not include the parameter.

1.12.2 SEPA Direct Debit - Merchant Supplied Mandate

If you elect to produce and provide your own Mandate to the consumer, the order flow differs from the Vantiv supplied Mandate flow, as shown in [Figure 1-10](#) and explained in the steps that follow.

FIGURE 1-10 Order Flow for Merchant Supplied Mandate



1. On your checkout page, the consumer selects SEPA Direct Debit as the payment method, enters their IBAN (International Banking Account Number), selects a preferred language for the Mandate (optional), and clicks the Submit button.
2. You redirect the consumer from your checkout page to your Mandate page.
3. The consumer take action on the Mandate page.
 - a. The consumer either accepts or rejects the Mandate.
 - b. Your Mandate page provides the Accept or Decline information to your servers.
4. Once the consumer takes action on the Mandate page, you return them to your checkout page. If the consumer rejected the Mandate, you can ask for a different payment method. If the consumer accepted the Mandate, you record the signature date and use it to construct a Sale transaction. Also, if this is the first of a series of recurring payments, you assign a mandateReference value.
5. You submit the Sale transaction to the Vantiv eComm platform, as shown in the example below. In addition to the required child elements of sepaDirectDebit you must include the mandateReference, mandateUrl, and mandateSignatureDate, as well as the name, email, and country child elements of billToAddress.

Example: SEPA Direct Debit Sale Request Message - Merchant Supplied Mandate

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
  merchantId="SddMer1">
  <authentication>
    <user>SDD</user>
    <password>password</password>
  </authentication>
  <sale id="SEPA_Sale" reportGroup="SddMer1" customerId="SEPA_Customer1">
    <orderId>sddSale</orderId>
    <amount>1001</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
      <name>Johann Schmidt</name>
      <addressLine1>10 Hoch Strasse</addressLine1>
      <city>Hanover</city>
      <zip>30159</zip>
      <country>DE</country>
      <email>jschmidt@phoenixprocessing.com</email>
      <phone>781-270-1111</phone>
    </billToAddress>
    <sepaDirectDebit>
      <mandateProvider>Merchant</mandateProvider>
      <sequenceType>FirstRecurring</sequenceType>
      <mandateReference>ABCD1234</mandateReference>
    </sepaDirectDebit>
  </sale>
</littleOnlineRequest>
```

```
<mandateUrl>http://MerchantMandateHostURL</mandateUrl>
<mandateSignatureDate>2017-01-21</mandateSignatureDate>
<iban>DE79850503003100180568</iban>
</sepaDirectDebit>
</sale>
</littleOnlineRequest>
```

6. The Vantiv eComm platform returns a Sales response message. This message does not include the `sepaDirectDebitResponse` element, since Vantiv does not return any additional SEPA Direct Debit related information, such as a `redirectToken`.

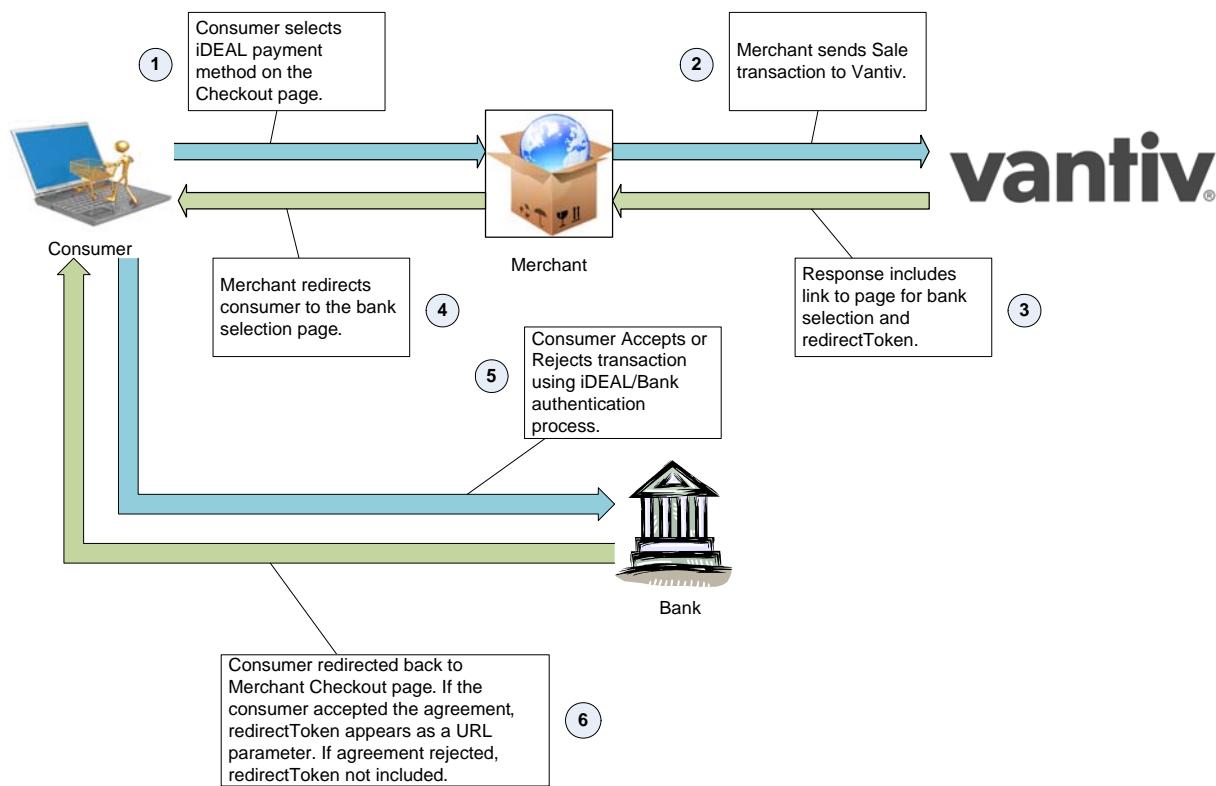
1.13 iDEAL

iDEAL is an International alternative payment method, used by over 10 million consumers in the Netherlands, that initiates a real-time transfer of funds from the consumer account to the merchant account. This method differs from credit cards and several other alternate payment methods in that the consumer is not agreeing to pay the settlement at a future time, rather they communicate directly with their bank and approve the transfer of fund from their account to yours. After their approval, their bank displays to them a confirmation of payment and debits their account within a few seconds.

NOTE: **Although the bank debits the consumer's account immediately, the funds may not appear in your account for a few days, depending upon your settlement agreement.**

The graphic and steps below illustrate the iDEAL process.

FIGURE 1-11 iDEAL Order Flow



1. On your checkout page, the consumer selects iDEAL as the payment method and clicks the Submit button.

2. You create a Sale transaction (example below) and submit it to the Vantiv eComm platform, using ideal as the payment method and including its child elements, preferredLanguage if specified.

Example: iDEAL Sale Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
  merchantId="12345">
  <authentication>
    <user>user_name</user>
    <password>password</password>
  </authentication>
  <sale id="TC6900_1" reportGroup="Mer1" customerId="pproPosResponse">
    <orderId>p1_idealSale</orderId>
    <amount>10011</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
      <name>David Berman</name>
      <addressLine1>10 Noorderstraat</addressLine1>
      <city>Amsterdam</city>
      <zip>1000 AP</zip>
      <country>NL</country>
      <email>dberman@phoenixprocessing.com</email>
      <phone>020-1234567</phone>
    </billToAddress>
    <ideal>
      <preferredLanguage>NL</preferredLanguage>
    </ideal>
  </sale></littleOnlineRequest>
```

3. The saleResponse includes a redirectURL, which you use to redirect the consumer to a bank selection page. It also contains a redirectToken, which you use later to verify the acceptance and bank transfer by the consumer.

Example: iDEAL Sale Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <saleResponse id="TC6900_1" reportGroup="PayProMer1">
    <littleTxnId>84568456</littleTxnId>
    <orderId>p1_idealSale</orderId>
    <response>000</response>
    <responseTime>2017-04-03T10:24:31</responseTime>
    <postDate>2017-04-03</postDate>
    <message>Approved</message>
```

```
<idealResponse>
  <redirectUrl>http://ideal.bank.com</redirectUrl>
  <redirectToken>1234567890</redirectToken>
  <paymentPurpose>123456YourCompanyName</paymentPurpose>
</idealResponse>
</saleResponse>
</littleOnlineResponse>
```

4. The consumer selects their bank on the bank selection page.
5. The consumer accepts or rejects the transaction on their bank page. The bank determines the actual process of accepting the transaction.
6. The consumer gets redirected back to your checkout page. If they accepted the agreement, the URL contains a parameter exposing the redirectToken value (redirectToken=token_value). By comparing this value to the value you received in the Sale response message you can verify that the consumer accepted the agreement and the bank transferred the funds. If they declined, the URL does not include the parameter.

NOTE: **Although the consumer does not have chargeback rights when using the iDEAL payment method, Vantiv recommends you maintain a robust and well documented returns system. European regulations require that you credit your customer's account within 30 days of a return.**

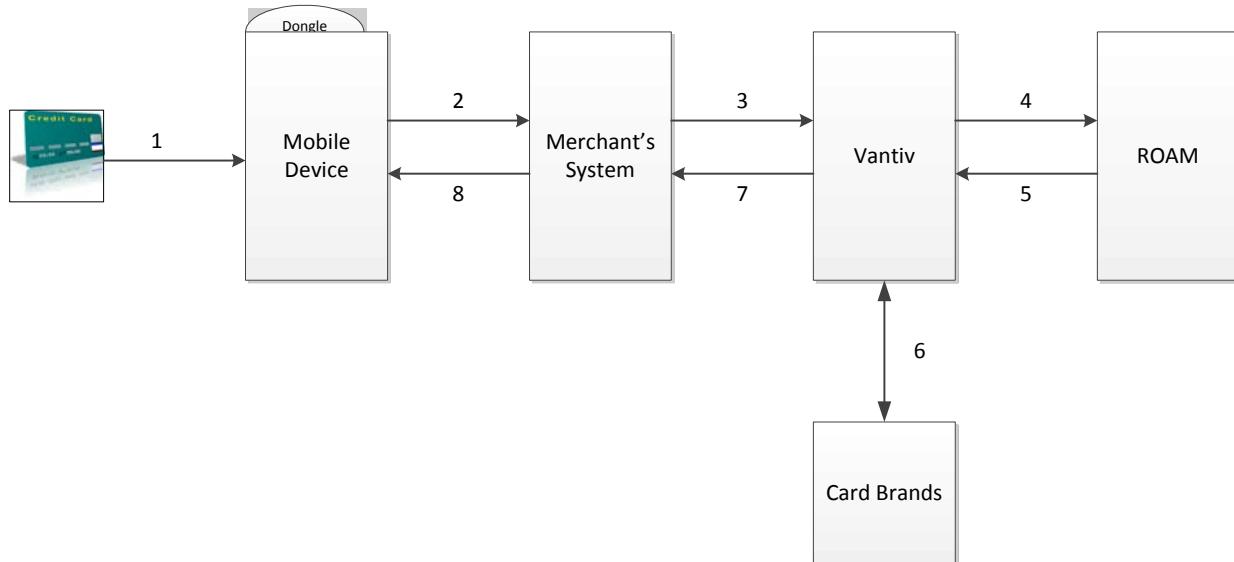
1.14 Mobile Point of Sale

In order to allow you to take advantage of the emerging area of mobile payments, Vantiv has partnered with ROAM™ to offer you a robust mobile solution. This section provides an overview of the transaction flow.

The illustration below shows an overview of the Authorization/Sale transaction flow.

1. Consumer supplied credit card is swiped through the dongle-coupled mobile device.
2. The mobile device sends the encrypted card information or the Auth/Sale transaction upstream to the merchant's systems.
3. The Merchant sends the Auth/Sale transaction, including the encrypted card information upstream to Vantiv.
4. Vantiv makes an API call to ROAM, submitting the encrypted card information for decryption.
5. ROAM returns the decrypted card information to Vantiv.
6. Vantiv submits the Auth/Sale transaction to the Card Brands for approval.
7. Vantiv returns the response file to the PayFac.
8. PayFac sends the transaction response to the mobile device.

FIGURE 1-12 Transaction Flow



1.15 eCommerce Solution for Apple Pay™

Vantiv supports Apple Pay for in-app and in-store purchases initiated on compatible versions of iPhone and iPad, as well as purchases from your desktop or mobile website initiated from compatible versions of iPhone, iPad, Apple Watch, MacBook and iMac (Apple Pay on the Web).

For in-store transactions, a consumer can use the Near Field Communications (NFC) chip in their device to make a purchase by simply touching the device to an NFC-compliant terminal. Identity verification is provided by Touch ID, a fingerprint reading application built into the device. If you wish to allow Apple Pay transactions from your native iOS mobile application, you must build the capability to make secure purchases using Apple Pay into your mobile application.

If you wish to allow Apple Pay payments on your desktop or mobile website, your website must be configured to work with Apple to request authorization from the consumers iPhone or iPad via Touch ID. See [Table 1-9, "Apple Pay on the Web Compatible Devices"](#) for more information.

This section provides an overview of the operation of Apple Pay and Apple Pay on the web, along with the several methods you can use to submit Apple Pay purchases to the Vantiv eCommerce platform. The topics discussed are:

- Overview of Apple Pay Operation
- Vantiv Decryption of Apple Pay PKPaymentToken
- Merchant Decryption of Apple Pay PKPaymentToken
- LittleXML <applepay> Structure
- Vantiv Mobile API for Apple Pay HTTPS POST Components
- Recurring Payments with Apple Pay

1.15.1 Overview of Apple Pay Operation

The operation of Apple Pay and Apple Pay on the web is relatively simple, but will require either the development of new native iOS applications or the modification of your existing applications or website that include the use of the Apple PassKit Framework (or Apple Pay JavaScript for Apple Pay web) and the handling of the encrypted data returned to your application by Apple Pay. The basic steps that occur when a consumer initiates an Apple Pay purchase using your mobile application or website are:

1. When the consumer selects the Apple Pay option from your application, your application makes use of the Apple PassKit Framework (or Apple Pay JavaScript) to request payment data from Apple Pay.
2. When Apple Pay receives the call from your application and after the consumer approves the Payment Sheet (using Touch ID), Apple creates a PKPaymentToken using your public key. Included in the PKPaymentToken is a network (Visa, MasterCard, Discover, or American Express) payment token and a cryptogram.

3. Apple Pay returns the Apple PKPaymentToken (defined in Apple documentation; please refer to <https://developer.apple.com/library/ios/documentation/PassKit/Reference/PaymentTokenJSON/PaymentTokenJSON.html>).

The remainder of this section discusses the various options for handling the PKPaymentToken in the transaction flow.

1.15.2 Vantiv Decryption of Apple Pay PKPaymentToken

Vantiv recommends using one of the Vantiv Decryption methods. This transaction process relieves you from the burden of creating and maintaining public and private keys, as well as decrypting the PKPaymentToken. If you have already implemented eProtect in a mobile application, you should use eProtect for Apple Pay. A second, similar method, which still allows you to submit the PKPaymentToken without decryption, involves you sending the Authorization/Sale transaction with the PKPaymentToken key values in a new LittleXML element structure (see [applepay](#)), typically from your server. This method can be used even if you are not tokenized with Vantiv.

In all of these implementations, your Vantiv Implementation Consultant will provide a CSR (Certificate Signing Request) you use in your registration process with Apple Pay. The CSR provides Apple Pay with the public key used for encryption, while Vantiv retains the private key used for decryption.

1.15.2.1 Using the Browser JavaScript API for Apple Pay on the Web

In this scenario, the Vantiv eProtect Customer Browser JavaScript API controls the fields on your checkout page that hold sensitive card data. When the cardholder clicks the Apple Pay button, communication is exchanged with Apple Pay via the JavaScript API to obtain the PKPaymentToken. From this point forward, your handling of the transaction is identical to any other eProtect transaction. The eProtect server returns a Registration ID (low-value token) and your server constructs the LittleXML transaction using that ID. See the *Vantiv eProtect Integration Guide* for JavaScript and HTML page examples and more information on using the browser JavaScript API.

[Step 1](#), [Step 2](#), and [Step 3](#) are the same as those outlined in [Overview of Apple Pay Operation](#) on page 59. The process after Step 3 is detailed below (and shown in [Figure 1-13](#)):

4. Your website sends the PKPaymentToken to our secure server via the JavaScript Browser API and eProtect returns a Registration ID.
5. Your website forwards the transaction data along with the Registration ID to your order processing server, as it would with any eProtect transaction.
6. Your server constructs/submits a standard LittleXML Authorization/Sale transaction using the Registration ID, setting the <orderSource> element to `applepay`.

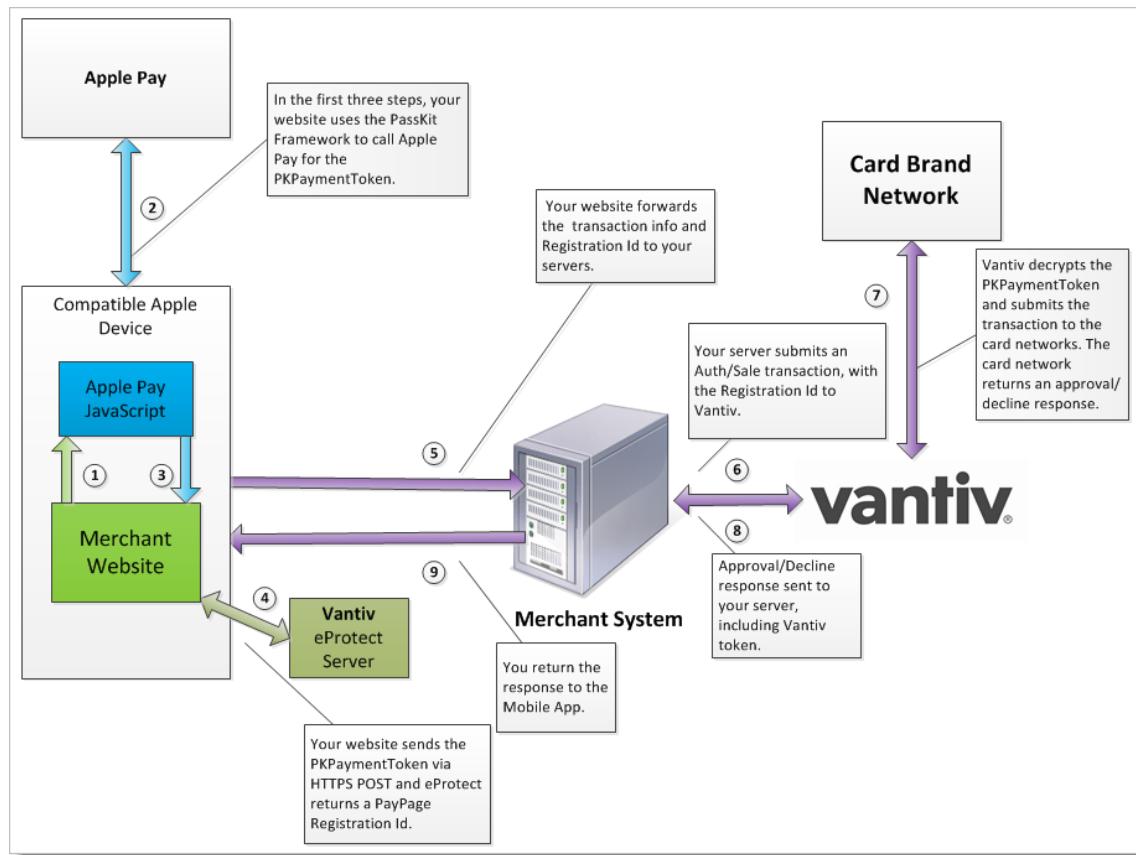
7. Using the private key, Vantiv decrypts the PKPaymentToken associated with the Registration ID and submits the transaction with the appropriate information to the card networks for approval.
8. Vantiv sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response and includes the Vantiv token.
9. You return the Approval/Decline message to your website.

Table 1-9 lists the requirements for your customers' Apple devices when making purchases via Apple Pay on the Web.

NOTE: **Table 1-9 represents data available at the time of publication of this document, and is subject to change. See the latest Apple documentation for current information.**

TABLE 1-9 Apple Pay on the Web Compatible Devices

Apple Device	Operating System	Browser
iPhone 6 and later iPhone SE	iOS 10 and later	
iPad Pro iPad Air 2 and later iPad Mini 3 and later	iOS 10 and later	
Apple Watch <i>Paired with iPhone 6 and later</i>	Watch OS 3 and later	Safari only
iMac <i>Paired with any of the above mobile devices with ID Touch for authentication</i>	macOS Sierra and later	
MacBook <i>Paired with any of the above mobile devices with ID Touch for authentication</i>	macOS Sierra and later	

FIGURE 1-13 Data/Transaction Flow using Browser JavaScript API for Apple Pay on the Web

1.15.2.2 Using the Vantiv Mobile API for Apple Pay

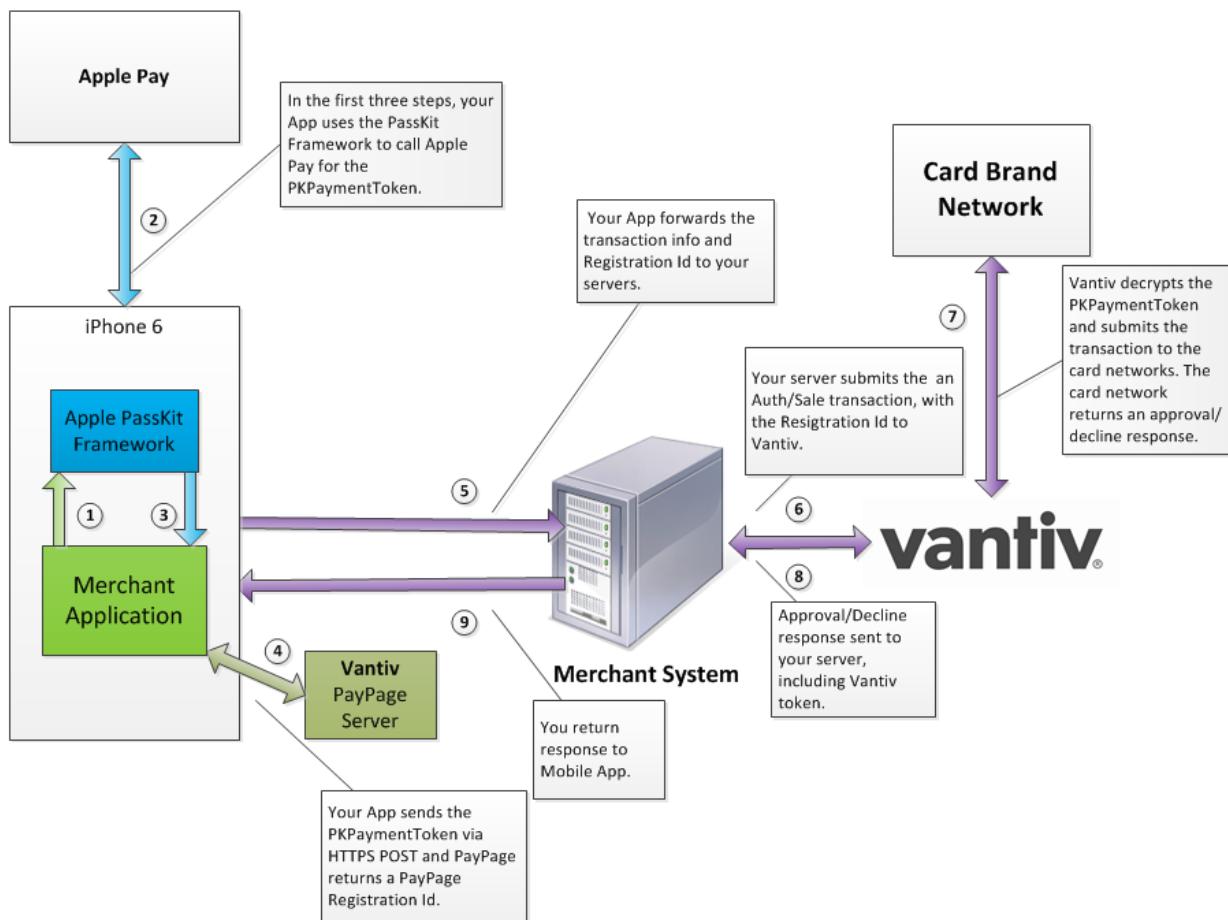
In this scenario, your native iOS application performs an HTTPS POST of the Apple Pay PKPaymentToken using the Vantiv Mobile API for Apple Pay. From this point forward, your handling of the transaction is identical to any other PayPage transaction. The PayPage server returns a PayPage Registration ID and your Mobile App (or server) constructs the LittleXML transaction using that ID.

Step 1, Step 2, and Step 3 are the same as those outlined in [Overview of Apple Pay Operation](#) on page 59. The process after Step 3 is detailed below and in [Figure 1-14](#).

4. Your native iOS application sends the PKPaymentToken to our secure server via an HTTPS POST and eProtect returns a Registration ID. Please refer to the *Vantiv eProtect Integration Guide* for additional information.
5. Your native iOS application forwards the transaction data along with the Registration ID to your order processing server, as it would with any eProtect transaction.
6. Your server constructs/submits a standard LittleXML Authorization/Sale transaction using the Registration ID, setting the `<orderSource>` element to `applepay`.

7. Using the private key, Vantiv decrypts the PKPaymentToken associated with the Registration ID and submits the transaction with the appropriate information to the card networks for approval.
8. Vantiv sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response and includes the Vantiv token.
9. You return the Approval/Decline message to your mobile application.

FIGURE 1-14 Data/Transaction Flow using the Vantiv Mobile API for Apple Pay



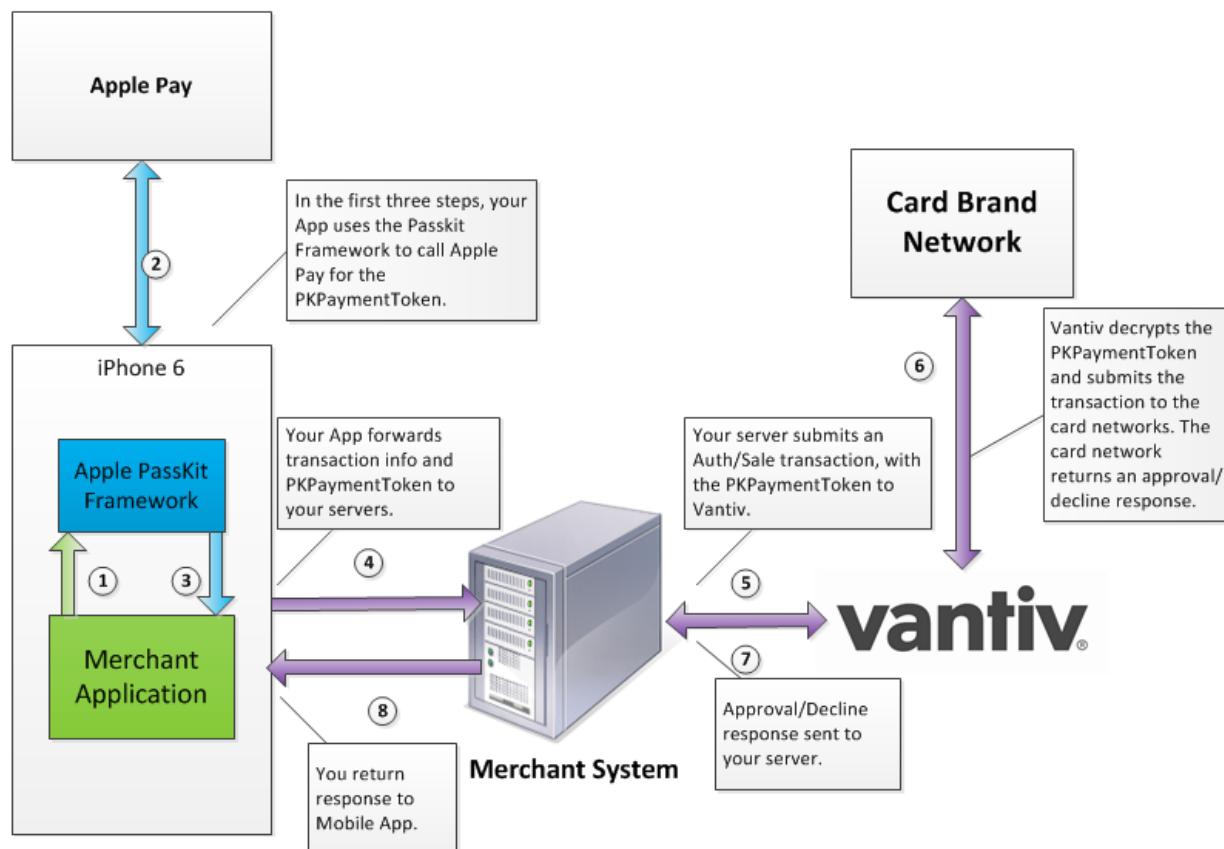
1.15.2.3 Submitting the Apple Pay PKPaymentToken in LittleXML

In this scenario, you submit the Apple Pay PKPaymentToken to the Vantiv eCommerce platform using a new structure in LittleXML (see [applepay](#)), typically from your server. As with the previous scenario, Vantiv decrypts the PKPaymentToken from Apple Pay using the private key.

Step 1, Step 2, and Step 3 are the same as those outlined in [Overview of Apple Pay Operation](#) on page 59. The process after Step 3 is detailed below and in [Figure 1-15](#).

4. Your mobile application forwards the PKPaymentToken from Apple Pay, along with other normal information from the transaction (such as Bill To and Ship To Address), to your order processing server.
5. You do not decrypt the PKPaymentToken, but rather forward the data to Vantiv in the Authorization/Sale transaction using the LittleXML <applepay> element structure instead of <card> (Server-side API submit) and setting the <orderSource> element to applepay.
6. Using the private key retained by Vantiv, we decrypt the PKPaymentToken and submit the transaction with the appropriate information to the card networks for approval.
7. Vantiv sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response.
8. You return the Approval/Decline message to your mobile application.

FIGURE 1-15 Data/Transaction Flow with Direct Submission of Apple Pay PKPaymentToken via LittleXML



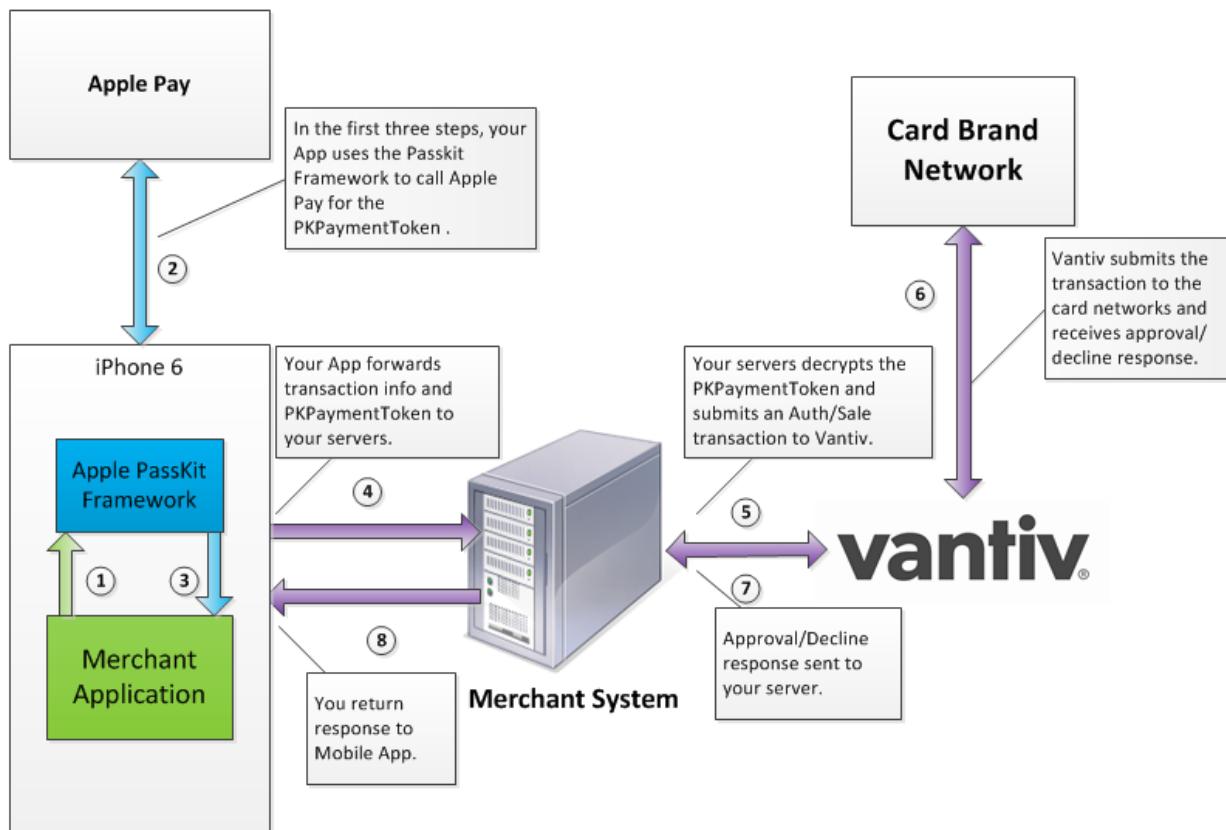
1.15.2.4 Merchant Decryption of Apple Pay PKPaymentToken

Using this process, the responsibility for the decryption of the PKPaymentToken from Apple Pay falls to you. After completing the first three steps of the process as detailed in the [Overview of](#)

Apple Pay Operation section and depicted by the green and blue arrows in [Figure 1-16](#), the process continues as follows:

4. Your mobile application forwards the PKPaymentToken from Apple Pay, along with other normal information from the transaction (such as Bill To and Ship To Address), to your order processing server.
5. Using your private key, you decrypt the PKPaymentToken, construct the Authorization/Sale transaction, and submit it to Vantiv. In this case, you would populate the LittleXML <number> element with the device primary account number, the <expDate> element with the expiration date, and the <authenticationValue> field with the cryptogram extracted from the PKPaymentToken. Also, set the <orderSource> element to applepay (Server-side API submit).
6. Vantiv detects that this is an Apple Pay transaction and submits the transaction with the appropriate information to the card networks for approval.
7. Vantiv sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response.
8. You return the Approval/Decline message to your mobile application.

FIGURE 1-16 Data/Transaction Flow with Merchant Decryption of Apple Pay PKPaymentToken



1.15.3 Recurring Payments with Apple Pay

When you submit the first transaction in a recurring/installment stream, you must set the `<processingType>` element to either **initialRecurring** or **initialInstallment**, as applicable. If the transaction involves a Visa or Discover card, the XML response message includes the `<networkTransactionId>` element. You must retain the value returned for use in future transactions. When you submit the next and all subsequent transactions in the recurring stream, set the `<orderSource>` to recurring and include the `networkTransactionId` value in the `<originalNetworkTransactionId>` element. If it is a Discover transaction also include the `<originalTransactionAmount>` element. Since the original payment token was only for a single use and is not available for additional transactions, this allows Visa to tie the new transaction to the original approved transaction.

1.16 eCommerce Solution for Android Pay™

Android Pay is an in-store and in-app (mobile or web) payment method, providing a secure process for consumers to purchase goods and services. In-store purchases are done by using Near Field Communication (NFC) technology built into the Android Smart phone with any NFC-enabled terminal at the retail POS. For in-app purchases, the consumer need only select Android Pay as the payment method in your application. You will need to modify your application to use Android Pay as a payment method.

Vantiv supports two methods for merchants to submit Android Pay transactions from Web/Mobile applications to the eComm platform. The preferred method involves you sending certain Vantiv specific parameters to Google®. The response from Google includes a Vantiv paypageRegistrationId, which you use in your payment transaction submission to Vantiv. With the alternate method, you receive encrypted information from Google, decrypt it on your servers, and submit the information to Vantiv in a payment transaction.

This section provides an overview of both methods and includes the following sections:

- [Android Pay using eProtect](#)
- [Merchant Decryption Method](#)

1.16.1 Android Pay using eProtect

This is the recommended and typical method of implementing Android Pay for Web and Mobile Applications on the Vantiv eComm platform. The steps that follow, along with [Figure 1-17](#), illustrate the high level flow of messages associated with an Android Pay purchase, when utilizing the Vantiv eProtect service.

NOTE:

This process assumes you have integrated with Google using the method that returns the Vantiv low-value token (paypageRegistrationId) from Google following the Full Wallet request.

1. When the consumer clicks the Android Pay button in your application, the action triggers a MaskedWalletRequest to Google. In the MaskedWalletRequest, you must set a new object PaymentMethodTokenizationParameters indicating that you are using Vantiv. Use the following code sample as a guide to setting this field.

Setting the PaymentMethodTokenizationParameters

```
PaymentMethodTokenizationParameters parameters =  
    PaymentMethodTokenizationParameters .newBuilder()  
        .setPaymentMethodTokenizationType(PaymentMethodTokenizationType.PAYMENT_GATEWAY)  
        .addParameter( "gateway" , "vantiv" )
```

```
.addParameter("vantiv:merchantPayPageId", payPageId)
.addParameter("vantiv:merchantOrderId", orderId)
.addParameter("vantiv:merchantTransactionId", id)
.addParameter("vantiv:merchantReportGroup", reportGroup)
.build();
```

IMPORTANT: You must use the same `orderId` value on all calls (i.e., Google, Register Token, Authorization, Sale, etc.). Failure to use the same `orderId` can prevent customers from tracking their orders using the Android Pay application.

Setting New Object in the MaskedWalletRequest

```
MaskedWalletRequest request = MaskedWalletRequest.newBuilder()
.setMerchantName(Constants.MERCHANT_NAME)
.setPhoneNumberRequired(true)
.setShippingAddressRequired(true)
.setCurrencyCode(Constants.CURRENCY_CODE_USD)
.setEstimatedTotalPrice(cartTotal)
.setCart(Car.newBuilder())
.setCurrencyCode(Constants.CURRENCY_CODE_USD)
.setTotalPrice(cartTotal)
.setLineItems(lineItems)
.build()
.setPaymentMethodTokenizationParameters(parameters)
.build();
```

The information returned by Google in the MaskedWallet object may include a masked card number (last-four digits exposed) and shipping information. The consumer has the option of changing this information. If any info changes, Android Pay returns an updated MaskedWallet object.

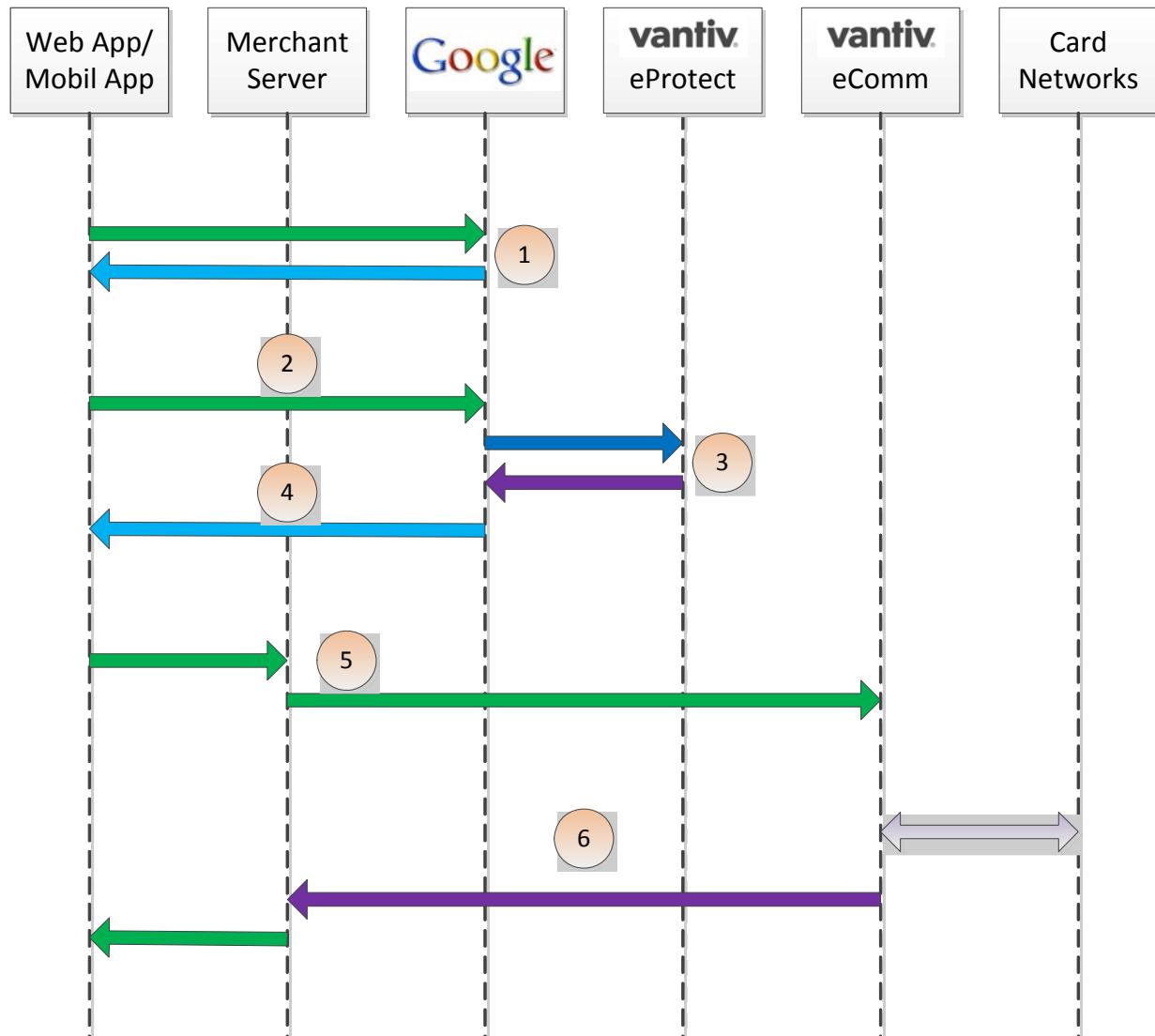
2. Upon confirmation of the order by the consumer your application initiates a FullWalletRequest to Google.
3. After receiving the FullWalletRequest from your application, Google submits the card information to Vantiv eProtect. The eProtect servers return a low-value token (`paypageRegistrationId`).
4. Google returns the low-value token to your application along with the Full Wallet information.

5. Your application sends the transaction information to your servers along with the low-value token. Your servers submit the Auth/Sale transaction to the Vantiv eComm platform. You must set the `orderSource` to **androidpay** in the transaction.

NOTE: Instead of submitting a Auth/Sale transaction, you can submit a Register Token transaction to convert the low-value token to a Vantiv high-value token. You would then use the high-value token in subsequent transactions submitted to the eComm platform.

6. Vantiv processes your transaction normally and returns the results along with a high-value token.

FIGURE 1-17 High Level Message Flow for Android Pay using eProtect



1.16.2 Merchant Decryption Method

Using this process, the responsibility for the decryption of the encrypted payload from Android Pay falls to you. The steps that follow, along with [Figure 1-18](#), illustrate the high level flow of messages associated with an Android Pay purchase, when you perform the decryption of the encrypted payload.

NOTE:

The process assumes you have integrated with Google using the method that returns the encrypted payload from Google following the Full Wallet request.

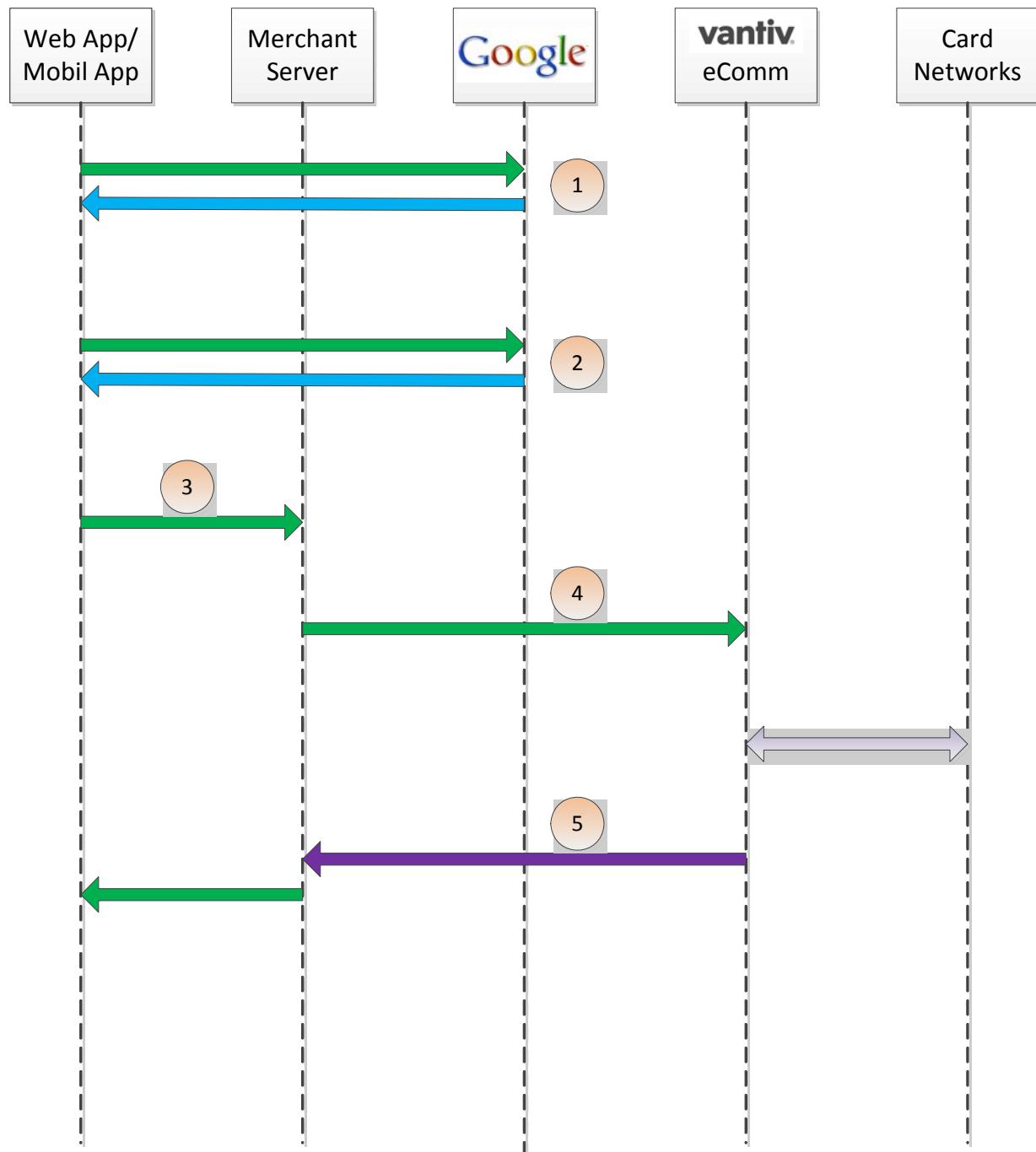
1. When the consumer clicks the Android Pay button in your application, the action triggers a MaskedWalletRequest to Google. The information returned by Google in the MaskedWallet object may include a masked card number (last-four digits exposed) and shipping information. The consumer has the option of changing this information. If any info changes, Android Pay returns an updated MaskedWallet object.
2. Upon confirmation of the order by the consumer your application initiates a FullWalletRequest to Google. Google also returns the encrypted payload. The encrypted payload is a UTF-8 encoded serialized JSON dictionary with the following keys:
 - **encryptedMessage** (string base64) - an encrypted message containing the payment credentials
 - **ephemeralPublicKey** (string base64) - the ephemeral public key associated with the private key to encrypt the message
 - **tag** (string base64) - MAC of encryptedMessage
3. Your application sends the encrypted payload along with the transaction information to your server.
4. Your server decrypts the encrypted payload extracting the payment, which is a UTF-8 encoded, serialized JSON dictionary with the following keys:
 - **dpan** (string (digits only)) - the device-specific personal account number (i.e., device token)
 - **expirationMonth** (number) - the expiration month of the dpan (1 = January, 2 = February, etc.)
 - **expirationYear** (number) - The four-digit expiration year of the dpan (e.g., 2015)
 - **authMethod** (string) - the constant **3DS** (may change in future releases).
 - **3dsCryptogram** (string) - the 3DSecure cryptogram
 - **3dsEciIndicator** ((optional) string) - ECI indicator per 3DSecure specification

Example of Decrypted Credentials in JSON

```
{  
    "dpan": "4444444444444444",  
    "expirationMonth": 10,  
    "expirationYear": 2015,  
    "authMethod": "3DS",  
    "3dsCryptogram": "AAAAAA...",  
    "3dsEciIndicator": "eci indicator"  
}
```

After decryption, submit the Authorization/Sale transaction to Vantiv, setting the `orderSource` element to **androidpay** and populating the following LittleXML elements with the decrypted information:

- **number** - dpan value
 - **expDate** - MMYY derived from the expirationMonth and expirationYear values
 - **authenticationValue** - the 3dsCryptogram value
5. Vantiv processes your transaction normally and returns the results in the response message.

FIGURE 1-18 High Level Message Flow for Android Pay using Merchant Decryption

1.16.3 Recurring Payments with Android Pay

When you submit the first transaction in a recurring/installment stream, you must set the `<processingType>` element to either **initialRecurring** or **initialInstallment**, as applicable. If the transaction involves a Visa or Discover card, the XML response message includes the `<networkTransactionId>` element. You must retain the value returned for use in future transactions. When you submit the next and all subsequent transactions in the recurring stream, set the `<orderSource>` to recurring and include the `networkTransactionId` value in the `<originalNetworkTransactionId>` element. If it is a Discover transaction also include the `<originalTransactionAmount>` element. Since the original payment token was only for a single use and is not available for additional transactions, this allows Visa to tie the new transaction to the original approved transaction.

1.17 Supported Transaction Types

LittleXML Batch processing supports all transaction types except Voids, eCheck Voids and Private Label Gift Card reversal transactions, 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.

1.17.1 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:	To obtain better interchange rates, you should always perform an AVS check either as a standalone transaction, or as part of the Authorization/Sale transaction.
	If you settle in a currency other than US or Canada, you cannot use partial Authorizations.
	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 1-10](#) provides information concerning Authorization lifespans for various card types and alternate payment methods.

TABLE 1-10 Lifespan of Payment Authorizations

Payment Type	Lifespan of Authorization
MasterCard	7 days
Visa	7 days
American Express	7 days
Discover	10 days

TABLE 1-10 Lifespan of Payment Authorizations

Payment Type	Lifespan of Authorization
PayPal	29 days total by default; Vantiv recommends capture submission within three days. For more information, see the <i>Vantiv PayPal 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 78](#)).

1.17.1.1 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.

NOTE: To obtain better interchange rates, you should always perform an AVS check either as a standalone transaction, or as part of the Authorization/Sale transaction.

1.17.2 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 any misuse of Auth fees imposed by the card associations.

NOTE: For American Express transactions, the reversal amount must match the authorization amount. American Express does not allow partial reversals and reversals against remaining amounts after a partial capture. 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, Vantiv automatically performs a partial Authorization Reversal for the unused amount (also see [Capture Request](#) on page 231).

1.17.2.1 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 use LittleXML format.
- Vantiv 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 associate 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, Vantiv 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, Vantiv 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 littleTxnId* 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.
- Do not use an Authorization Reversal to reverse an Authorization on a (Closed Loop) Gift Card. Use a Gift Card Auth Reversal instead (see [Gift Card Auth Reversal Transactions](#) on page 290).

1.17.2.2 Using Authorization Reversal to Halt Recycling Engine

If you are using the 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.

1.17.3 Activate Transaction

You use an Activate transaction to change the status of a (Closed Loop) Gift Card from an inactive to an active state with a value as specified by the `amount` element. You can also use this transaction type to create a Virtual Gift Card.

1.17.4 Activate Reversal Transaction (Online Only)

You use an Activate Reversal transaction to change the status of a newly activated (Closed Loop) Gift Card from active to inactive, thus reversing the operation of an Active transaction. The Activate Reversal references the associated Activate transaction by means of the `littleTxnId` element returned in the Activate response.

1.17.5 Balance Inquiry Transaction

You use the Balance Inquiry transaction to determine the balance available for use on a (Closed Loop) Gift Card.

1.17.6 Cancel Subscription Transaction

If you are using the Recurring Engine, you create Subscriptions as part of an Authorization or Sale transaction. You use the Cancel Subscription transaction to cancel the specified subscription, removing the transaction stream managed by the Recurring Engine.

1.17.7 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: If you settle in a currency other than US or Canada, you cannot use partial captures.

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

Do not use a Capture transaction to capture funds on a (Closed Loop) Gift Card transaction; use a Gift Card Capture instead (see [Gift Card Capture Transactions](#) on page 292).

1.17.8 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 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 system, but the `littletxnId` is unknown by the submitting party (for example, if the Auth was submitted by a merchant, but a fulfills submits a Capture Given Auth).

Whenever you submit a Capture Given Auth transaction, Vantiv 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 Vantiv 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 Vantiv will issue an Auth Reversal transaction for the balance of the Authorization. This is done to obtain the best possible interchange rates from Visa.

1.17.9 Create Plan Transaction

You use the Create Plan transaction to define several attributes of a recurring payment schedule. Later, as part of an Authorization or sale transaction, you can associate existing, active plans with Subscriptions. This association establishes a recurring payment schedule managed by the Vantiv Recurring Engine.

1.17.10 Credit Transaction

You use a Credit transaction to refund money to a customer, even if the original transaction occurred outside of the 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: Vantiv recommends you send all Credit transactions in a Batch separate from the associated Capture or Sale transactions.

1.17.11 Deactivate Transaction

You use a Deactivate transaction to change the status of a (Closed Loop) Gift Card from an active to an inactive state.

1.17.12 Deactivate Reversal Transaction (Online Only)

You use a Deactivate Reversal transaction to change the status of a newly deactivated Gift Card from inactive to active, thus reversing the operation of an Deactivate transaction. The Deactivate Reversal references the associated Deactivate transaction by means of the `littleTxnId` element returned in the Deactivate response.

1.17.13 Deposit Reversal Transaction (Online Only)

Used only for (Closed Loop) Gift Card related transactions, a Deposit Reversal transaction to reverse the funds capture initiate by either a Capture or Sale transaction. The Deposit Reversal references the associated Capture/Sale transaction by means of the `littleTxnId` element returned in the Capture/Sale response. You should never attempt to use this transaction type to reverse credit card or eCheck transactions.

1.17.14 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 system.

1.17.15 eCheck Prenotification Credit Transaction

You use this non-monetary transaction to verify the consumer's account information prior to submitting an eCheck Credit transaction (also see [eCheck Prenotification](#) on page 47). This transaction type is only supported for US transactions.

1.17.16 eCheck Prenotification Sale Transaction

You use this non-monetary transaction to verify the consumer's account information prior to submitting an eCheck Sale transaction (also see [eCheck Prenotification](#) on page 47). This transaction type is only supported for US transactions.

1.17.17 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.**

1.17.18 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.

1.17.19 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. This transaction type is only supported for US transactions.

1.17.20 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.

1.17.21 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 VANTIV BEFORE PROCESSING THIS TRANSACTION. IN SOME INSTANCES, USING A FORCE CAPTURE TRANSACTION CAN LEAD TO CHARGEBACKS AND FINES.**

1.17.22 Gift Card Auth Reversal

You use the Gift Card Auth Reversal transaction to reverse an Authorization on a (Closed-loop) Gift Card. This removes the hold on the previously authorized amount, freeing those funds for additional purchases by the cardholder.

1.17.23 Gift Card Capture

You use the Gift Card Capture transaction to capture or deposit previously authorized funds on a (Closed-loop) Gift Card.

1.17.24 Gift Card Credit

You use a Gift Card Credit transaction to refund money to a customer that used a (Closed Loop) Gift Card for the purchase, even if Vantiv did not process the purchase. You have a choice of two structures. You use the message type containing a transaction Id to apply a credit for a transaction processed by Vantiv. You use the message type containing a order Id to apply a credit for a transaction not processed by Vantiv.

1.17.25 Load Transaction

You use a Load transaction to add funds to an active Gift Card. The load amount cannot exceed the maximum allowed amount for the Gift Card. If you attempt to load more than the maximum amount, the transaction will be declined with a response Code of 221 - Over Max Balance.

1.17.26 Load Reversal Transaction (Online Only)

You use a Load Reversal transaction to reverse the operation of a Load transaction, removing the newly loaded amount from the Gift Card. You cannot perform a partial Load Reversal. This transaction always reverses the full amount of the referenced Load transaction.

1.17.27 Refund Reversal Transaction (Online Only)

The Refund Reversal transaction is a (Closed Loop) Gift Card only transaction that reverses the operation of a Refund transaction on the Gift Card. You cannot perform a partial Refund Reversal. This transaction always reverses the full amount of the referenced Refund transaction.

1.17.28 Register Token Transaction

You use the Register Token transaction to submit a credit card number, eCheck account number, or Registration Id to us 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 Batches and replace them in your database with the tokens returned.

1.17.29 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.**

To obtain better interchange rates, you should always perform an AVS check either as a standalone transaction, or as part of the Authorization/Sale transaction.

1.17.30 Status Query Transaction

The Status Query Transaction allows you to verify that an Online transaction submitted within the prior 24 hours exists in the system. The response will be one of the following:

- A single transaction matching the search criteria
- Multiple transactions matching the search criteria
- Empty results, if no transactions matched the criteria
- A limited response, if a transaction was found, but processing was not complete

As search criteria, you must submit, at a minimum, the id (id attribute) and transaction type (i.e., authorization, deposit, void, etc.) of the original transaction, but to narrow the search, you can also include the transaction id, order id, and the account number (credit, debit, or gift card) from the original transaction. The response message contains one of four response codes, 150 through 153 (see [Payment Transaction Response Codes](#) on page 812), and the results for the search.

NOTE: **Use of this transaction type requires specific permissions. Please speak to your Vantiv Implementation Consultant for additional information.**

1.17.31 Unload Transaction

You use an Unload transaction to remove funds from an active Gift Card. The unload amount cannot exceed the available balance on the Gift Card. If you attempt to unload more than the available balance, the transaction will be declined with a response Code of 209 - Invalid Amount.

1.17.32 Unload Reversal Transaction (Online Only)

The Unload Reversal transaction reverses the operation of a Unload transaction, returning the value removed from the Gift Card by the Unload transaction. You cannot perform a partial Unload Reversal. This transaction always reverses the full amount of the referenced Unload transaction.

1.17.33 Update Card Validation Number Transaction

When you submit the CVV2/CVC2/CID in a `registerTokenRequest`, the platform encrypts and stores the value on a temporary basis for later use in a tokenized Auth/Sale transaction submitted without the value. This is done to accommodate merchant systems/workflows where the security code is available at the time of token registration, but not at the time of the Auth/Sale. If for some reason you need to change the value of the security code supplied at the time of the token registration, use an `updateCardValidationNumOnToken` transaction.

1.17.34 Update Plan Transaction

You use the Update Plan transaction to activate/deactivate Plans associated with recurring payments. When you deactivate a Plan, by setting the `active` flag to `false`, you can no longer reference that Plan for use with subscriptions. Existing subscriptions already using the deactivated Plan will continue to use the Plan until the subscription is either modified or completed. You can also reactivate a deactivated Plan by updating the Plan and setting the `active` flag to `true`.

1.17.35 Update Subscription Transaction

You use an Update Subscription transaction to change certain subscription information associated with a recurring payment. Using this transaction type you can change the plan, card, billing information, and/or billing date. You can also create, update, or delete a Discount and/or an Add On.

1.17.36 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 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 75.)**

1.17.36.1 Using Void to Halt Recycling Engine

If you use Recovery or the Recycling Engine service and use 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.**

When using a Void transaction to halt recycling, there is a possibility that the recycled transaction has already been approved and captured. If this condition occurs, depending upon your configuration, the system takes one of two actions:

- If you are not configured for the Automatic Refund option (default = disabled), the system declines the Void transaction. You must issue the Credit transaction to refund the money to the consumer. The daily Recycling file will include the approved/captured transaction.
- If you are configured for the Automatic Refund option, the system issues a Credit transaction on your behalf. The system returns the transaction Id for the Credit transaction in the Void response message (`creditLittleTnxId` element). The daily Recycling file will include the approved/captured transaction only if the file was generated prior to the system receiving the Void and issuing the automatic refund.

1.17.37 Instruction-Based Dynamic Payout Transactions

If you are a PayFac using the Instruction-Based Dynamic Payout model, there are a number of Batch only transaction types you use to move funds between various accounts, including funding Sub-merchants. This section provides information about these transaction types. For additional information, please refer to the [Appendix D, "PayFac™ Dynamic Payout"](#).

- **Funding Instruction Void** - You use this transaction type to void a submitted funding instruction. You must submit the void prior to your daily cutoff time for an instruction not yet processed.
- **PayFac Credit Transaction** - You use this transaction type to move funds from the PayFac Settlement Account to your Operating Account.
- **PayFac Debit Transaction** - You use this transaction type to move funds from your Operating Account to the PayFac Settlement Account.

- **Physical Check Credit** - You use this transaction type to move funds from the PayFac Settlement Account to the account of a third party that issues physical checks on your behalf.
- **Physical Check Debit** - You use this transaction type to move funds from the physical check issuer's Account to the PayFac Settlement Account.
- **Reserve Credit Transaction** - You use this transaction type to move funds from the PayFac Settlement Account to the Reserve Account.
- **Reserve Debit Transaction** - You use this transaction type to move funds from the Reserve Account to the PayFac Settlement Account.
- **Sub-merchant Credit Transaction** - You use this transaction type to move funds from the PayFac Settlement Account to the Sub-merchant Account, funding the Sub-merchant.
- **Sub-merchant Debit Transaction** - You use this transaction type to move funds from the Sub-merchant Account to the PayFac Settlement Account.
- **Vendor Credit** - You use this transaction type to move funds from the PayFac Settlement Account to the account of a third party involved in the sale transactions, but are not the sub-merchant.
- **Vendor Debit** - You use this transaction type to move funds from the Vendor Account to the PayFac Settlement Account.

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 and Testing Environments
- Overview of Testing
- Transferring Files
- Performing the Required Certification Tests
- Performing the Optional Tests

IMPORTANT: Per PCI DSS Requirements and Security Assessment Procedure, Section 6.4.3, "Production data (live PANs) are not used for testing or development."

NOTE: This chapter does not include Certification tests for the PayFac Instruction-Based Dynamic Payout transaction types. Additional information about these transactions, as well as the Certification tests are in [Appendix D, "PayFac™ Dynamic Payout"](#).

2.1 Certification and Testing Environments

Vantiv has several testing and certification platforms optimized for different uses. These environments are called: Sandbox, Pre-Live, and Post-Live. This section discusses the various environments, their uses, and limitations. The certification and testing environments do not have the full capacity, performance, or availability of the Vantiv Production platform.

2.1.1 Sandbox Environment

The Sandbox environment is a simulator that provides a basic interface for functional level testing of transaction messages. Typically, merchants using one of the available Software Development Kits (SDKs) would use the Sandbox to test basic functionality, but it could also be used by any merchant using LittleXML. This is a stateless simulator, so it has no historical transactional data, and does not provide full functionality. This environment is not intended for certification or regression testing.

NOTE: At this time, the Sandbox does not support Batch transactions (Online transactions only).

Use of the Sandbox does not require a password. The environment is available on the public internet (at www.testtitle.com/sandbox/communicator/online). Supporting documentation, including test case documentation, is available at www.litle.com/developers/litle-sandbox.

2.1.2 Pre-Live Environment

The Pre-Live test environment is the platform used for all merchant Certification testing. This environment should be used by both newly on-boarding Vantiv merchants (for example, coding a direct XML integration for the first time), and existing Vantiv merchants seeking to incorporate additional features or functionalities into their current XML integrations. While the Pre-Live system provides a working version of the Vantiv production system, it does not have the full capacity or performance of the production platform and operates using simulators rather than communicating with the card associations.

The Pre-Live environment provides for the self-provisioning of a basic test account via www.testtitle.com/sandbox/. Using this account, you can submit most standard transaction types, including those specified in the Certification test sections provided later in this chapter. You will not be able to test many of the Vantiv eCommerce Value Added Services (VAS) using this basic account. To add the capability to perform test scenarios for the VAS services, please contact a Vantiv Implementation Consultant. They will adjust the configuration of your test account to enable the additional, desired features.

2.1.2.1 Pre-Live Environment Limitations and Maintenance Schedules

When using the Pre-Live environment for testing, please keep in mind the following limitations and maintenance schedules:

- The number of merchants configured per organization will be limited to the number necessary to perform the required certification testing.
- Data retention is limited to a maximum of 30 days.

NOTE: **Depending upon overall system capacity and/or system maintenance requirements, data purges may occur frequently. Whenever possible, we will provide advanced notification.**

- Merchant profile and data deleted after 7 consecutive days with no activity.
- Maintenance window - each Tuesday and Thursday from 4:00-8:00 AM ET.
- Daily limit of 1000 Online transactions.
- Daily limit of 10000 Batch transactions.

2.1.3 Post-Live Environment

The Post-Live testing environment is intended for merchants that are already fully certified and submitting transactions to the Vantiv production platform, but wish to perform regression or other tests to confirm the operational readiness of modifications to their own systems. Upon completion of the initial certification and on-boarding process, Vantiv migrates merchants that are Production-enabled to the Post-Live environment for any on-going testing needs.

In Post-Live, the merchant profile matches your profile from the Vantiv Production environment. Similar to Pre-Live, the Post-Live system provides a working version of the Vantiv production system, but it does not have the full capacity or performance of the production platform and operates using simulators rather than communicating with the card associations. Unlike the Pre-live platform, all Merchant accounts/IDs are synchronized with the Vantiv production environment.

2.1.3.1 Post-Live Environment Limitations and Maintenance Schedules

When using the Post-Live environment for testing, please keep in mind the following limitations and maintenance schedules:

- Daily limit of 1000 Online transactions.
- Daily limit of 10000 Batch transactions.
- Maintenance window - each Tuesday and Thursday from 4:00-8:00 AM ET.
- Data retention limited to a maximum of 30 days.

NOTE: Depending upon overall system capacity and/or system maintenance requirements, data purges may occur frequently. Whenever possible, advanced notification will be provided.

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 testing process involves submitting Vantiv supplied data for specific fields in a request, and receiving specific data back in a response. The response returned by Vantiv 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, Vantiv 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 offer many options/optional fields that you may wish to utilize 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 Insights feature set, there are additional Authorization tests required for certification.

NOTE:	For information about certification testing for PayPal, eProtect, Chargeback API, and the PayFac API, please refer to the following documents:
	– <i>Vantiv PayPal Integration Guide</i>
	– <i>Vantiv eProtect Integration Guide</i>
	– <i>Vantiv Chargeback API Reference Guide</i>
	– <i>Vantiv PayFac API Reference Guide</i>

2.2.2 Required Certification Testing

Certification testing is a required phase of implementing the LittleXML format. During the certification testing, a Little & Co. Implementation Consultant works with you to verify that your transaction submissions meet the requirements of the LittleXML specifications. Each transaction type has specific test scenarios that use specific data sets simulating real transactions. The Vantiv 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 101.

2.2.3 Optional Testing

Little & 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 and Testing Environments](#) on page 88). 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@vantiv.com.
--------------	---

2.2.4 System Doctor

The System Doctor is a tool you can use to both verify the operation of the Pre-Live environment and to verify operation and message structures of individual certification tests. Every 30 minutes the System Doctor runs each certification test specified in this chapter, except the Recycling and Recurring tests. You access the System Doctor through the Pre-Live iQ (Operations>System Doctor).

FIGURE 2-1 System Doctor

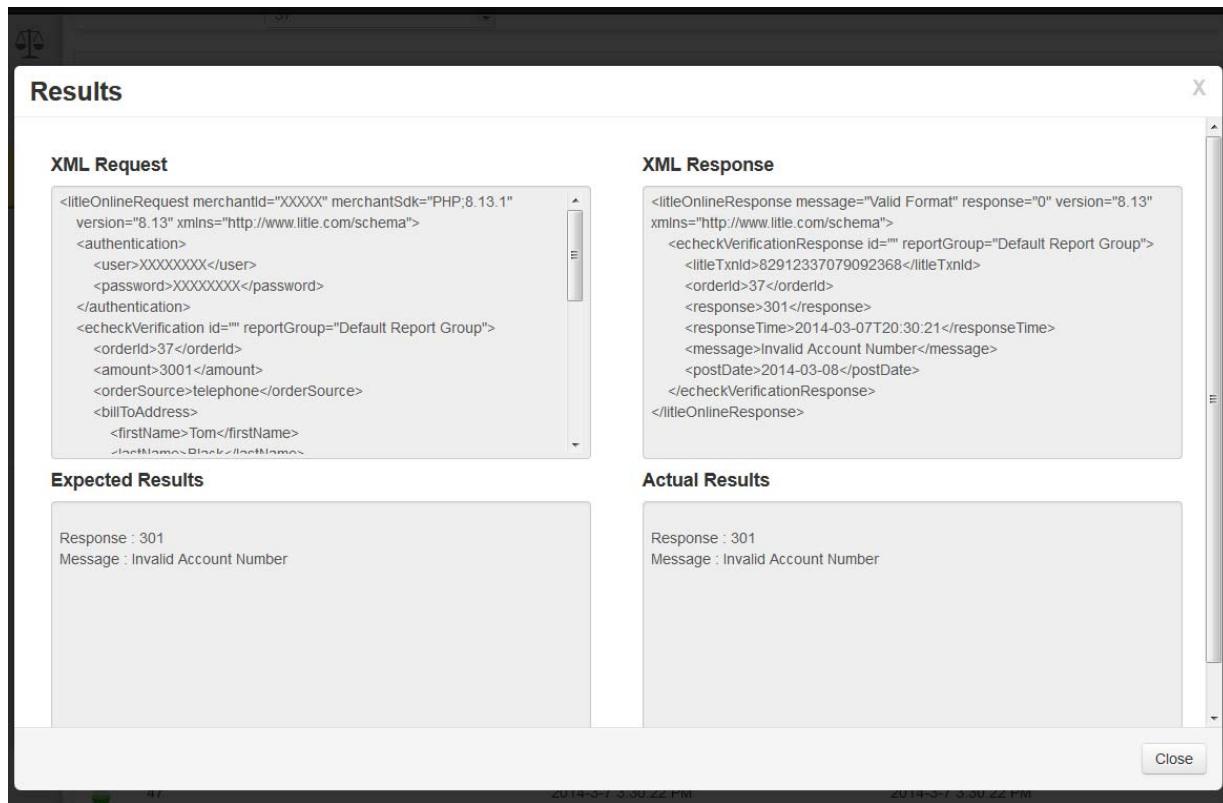
The screenshot shows the 'Systems Doctor' interface. At the top, there's a header bar with links like Transaction Search, Transaction Summary, Sessions, BIN Lookup, Authorization, Virtual Terminal, Presenter Credentials, Payfac Portal, and Systems Doctor. Below the header is a 'Overall Status' section with a cloud icon, showing 210 of 210 Transactions Successful, last updated on 2014-03-14 09:31:29 AM, and an average FTP Response Time of 80 seconds. To the right of this is a 'Search Tags' grid containing various payment-related terms. A dropdown menu titled 'Select Order Id' lists several order IDs: None, Payment Visibility, 1, 1A, and 37. Below these sections is a 'Test Execution History' table with columns for Time Stamp, Status, and Transaction Remarks. The table lists five rows of successful transactions from March 14, 2014.

Time Stamp	Status	Transaction Remarks
2014-3-14 9:31:29 AM	Success	210 / 210 Transactions Successful
2014-3-14 9:01:16 AM	Success	210 / 210 Transactions Successful
2014-3-14 8:31:11 AM	Success	210 / 210 Transactions Successful
2014-3-14 8:01:24 AM	Success	210 / 210 Transactions Successful
2014-3-14 7:31:15 AM	Success	210 / 210 Transactions Successful

As shown in [Figure 2-1](#), the system displays an overall assessment of the system status in the top panel and the results from each test set in the Test Execution History section (bottom panel). The center panel has a list of all individual tests by Order Id (pull-down), which you can filter by clicking on or more of the Search Tag buttons. Selecting one of the Order Ids from the list displays the XML request and response for that test (see [Figure 2-2](#)). You can also display the XML request and response information by selecting a test run from the Test Execution History and then selecting an individual test from the expanded list.

NOTE:

The search tags filter the Order Ids shown in the selection pull-down, but do so in an ORed manner. For example, if you select the Visa Filter and the Prepaid Filter, the pull-down displays Order Ids for tests that either use a Visa card OR a Prepaid card, rather than tests for a Prepaid Visa card.

FIGURE 2-2 XML Request and Response Messages

If one of your certification tests did not yield the expected results, you can use the System Doctor to determine if the test is performing as expected in the environment and if there is a discrepancy between your submitted XML and the XML message used in the automated test. If the test failed in the last automated run, you will know that there is a system issue and can contact your Implementation Consultant to determine when it will be resolved. If the test passed in the automated run, you can compare the automated XML message to your submission to determine why you are not receiving the expected results.

2.3 Transferring Files

As discussed in [Communications Protocols](#) on page 2, there are several protocols you can use to submit your transactions to Vantiv for processing. This section provides additional information concerning the recommended methods for transferring your LittleXML Batch and Online transaction files.

2.3.1 Transferring Session Files

This section describes how to FTP your files (not test system specific) and includes the following topics:

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

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

2.3.1.1 Submitting a Session File for Processing

CAUTION: When submitting a file via FTP/sFTP, verify that the file permission is set to 664.

Also, 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 Vantiv inbound directory and move your file to the Vantiv directory.
3. After the FTP process completes, change the extension of the transmitted file (in the Vantiv inbound directory) from .prg to .asc (lowercase). Also, change the file permission to 664. 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 Session Files

Depending on the size of your file, your response should be ready within a few minutes. Batches 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 acknowledgment 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: **Vantiv removes response files from the outbound directory after 24 hours.**
Plan to retrieve your files daily.

1. Open your FTP connection to the Vantiv 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, Vantiv 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 & "<littleOnlineRequest 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>0521</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 Vantiv 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 Vantiv and back to the merchant
2. Processing time by the card association or authorization provider
3. Processing time Vantiv

Under normal operating circumstances, the transmission time to and from Vantiv does not exceed 0.6 seconds and processing time by Vantiv 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, Vantiv 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 system.
If you elect to submit Online Sale transactions, Vantiv recommends a timeout setting of not less than 60 seconds to reduce the chances of undetected duplicate Sale transactions.

2.3.2.4 Notes on Persistent Connections

In order to provide a highly scalable service that meets the needs of high-throughput merchants, while reducing the number of idle connections that could result in some merchants exceeding their connection limits, Vantiv systems allow for 10 seconds of idle time before closing a connection. We selected this value because it is the midpoint between the Apache httpd 2.0 default value of 15 seconds and the Apache 2.2 default value of 5 seconds. Also, HTTP/1.1, which most modern HTTP client libraries use, employ persistent connections by default. When using a persistent connection, please keep the 10 second idle time limit in mind, when coding. If you do not use persistent connections, to avoid connection limit issues, please validate that your default configuration closes connections after each request.

NOTE: Proper use of persistent connections is considerably faster than opening and closing connections with each request.

2.3.2.5 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>
- <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 transactions to the Vantiv production system. This testing process allows you to verify that your system not only submits correctly formatted transaction data, but also correctly parses the data returned to you in the response messages. To facilitate the certification process, Vantiv has established a certification environment that simulates the production environment.

IMPORTANT: To determine the final status and response Code of a declined or duplicate transactions, please refer to the Declined Transaction Report in Vantiv iQ. Once in the production environment, you can also obtain this daily report via Secure Scheduled Reports.

During certification testing, a Vantiv Implementation Consultant will guide you through each required test scenario. For each transaction type specific data is supplied that you must use in your LittleXML 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.

NOTE: 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.
Never use the supplied test data in the production environment.

2.4.1 Testing Authorization (including Indicators), AVS Only, Capture, Credit, Sale, and Void Transactions

NOTE: To test your handling of 2-series (BIN) number for MasterCard, you can substitute a 2-series test card number (see [Appendix C, "Test Card Numbers"](#)) in any Auth/Sale test and you will receive a 000 - Approved Response Code in the response message.

[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 Vantiv 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 `littleTxnId` returned in the response messages.)
- Order Ids 10 through 13 - include if you plan to use **Partial Authorization**
- Order Ids 14 and 20 - include if you plan to use the **Prepaid Indicator** feature (see [Prepaid Indicator](#) on page 25)
- Order Ids 21 through 24 - include if you plan to use the **Affluence Indicator** feature (see [Affluence Indicator](#) on page 26)
- Order Id 25 - include if you plan to use the **Issuer Country** feature (see [Issuer Country Indicator](#) on page 26)

To test **Authorization** transactions:

1. Verify that your Authorization XML template is coded correctly (refer to [Authorization Transactions](#) on page 208.)
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 21through 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 25 of [Table 2-1](#). Verify that your response values match those shown in Key Response Elements for Order Id 25 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 314.)
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 231.).
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 247.)
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 334.)
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> 10100 <name> John & Mary Smith <addressLine1> 1 Main St. <city> Burlington <state> MA <zip> 01803-3747 <country> US <type> VI <number> 4457010000000009 <expDate> 0121 <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> 10100 <name> Mike J. Hammer <addressLine1> 2 Main St. <addressLine2> Apt. 222 <city> Riverside <state> RI <zip> 02915 <country> US <type> MC <number> 5112010000000003 <expDate> 0221 <cardValidationNum> 261 <authenticationValue> BwABBJQ1AgAAA AAgJDUAAAAAA A=		Authorization Response: <response> 000 <message> Approved <authCode> 22222 <avsResult> 10 <cardValidationResult> <authenticationResult> Note: Not returned for MasterCard	
2A	Capture: <littleTxnId>	Value returned in Auth response for Order Id 2	Capture Response: <response> 000 <message> Approved	
2B	Credit: <littleTxnId>	Value returned in Capture response for Order Id 2A	Credit Response: <response> 000 <message> 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> 000 <message> Approved	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
3	Authorization/Sale/AVS: <amount> 10100 <name> Eileen Jones <addressLine1> 3 Main St. <city> Bloomfield <state> CT <zip> 06002 <country> US <type> DI <number> 6011010000000003 <expDate> 0321 <cardValidationNum> 758		Authorization Response: <response> 000 <message> Approved <authCode> 33333 <avsResult> 10 <cardValidationResult> M	
3A	Capture: <littleTxnId> Value returned in Auth response for Order Id 3		Capture Response: <response> 000 <message> Approved	
3B	Credit: <littleTxnId> Value returned in Capture response for Order Id 3A		Credit Response: <response> 000 <message> 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> 000 <message> Approved	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
4	Authorization/Sale/AVS: <amount> 10100 <name> Bob Black <addressLine1> 4 Main St. <city> Laurel <state> MD <zip> 20708 <country> US <type> AX <number> 375001000000005 <expDate> 0421		Authorization Response: <response> 000 <message> Approved <authCode> 44444 <avsResult> 13	
4A	Capture: <littleTxnId> Value returned in Auth response for Order Id 4		Capture Response: <response> 000 <message> Approved	
4B	Credit: <littleTxnId> Value returned in Capture response for Order Id 4A		Credit Response: <response> 000 <message> 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> 000 <message> Approved	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
5	Authorization/Sale/AVS: <amount> 10100 <type> VI <number> 4100200300011001 <expDate> 0521 <cardValidationNum> 463 <authenticationValue> BwABBJQ1AgAAA AAgJDUCAAAAAA A=		Authorization Response: <response> 000 <message> Approved <authCode> 55555 <avsResult> 32 <cardValidationResult> M	
5A	Capture: <littleTxnId>	Value returned in Auth response for Order Id 5	Capture Response: <response> 000 <message> Approved	
5B	Credit: <littleTxnId>	Value returned in Capture response for Order Id 5A	Credit Response: <response> 000 <message> 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> 000 <message> Approved	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
6	Authorization/Sale: <amount> 10100 <name> Joe Green <addressLine1> 6 Main St. <city> Derry <state> NH <zip> 03038 <country> US <type> VI <number> 4457010100000008 <expDate> 0621 <cardValidationNum> 992		Authorization Response: <response> 110 <message> Insufficient Funds <avsResult> 34 <cardValidationResult> P	
6A	Void: <littleTxnId> Use the value returned in the Sale response (not Auth) for Order Id 6.		Void Response: <response> 000 <message> Approved Declined Transaction report result = 360 - No transaction found with specified littleTxnId	
7	Authorization/Sale/AVS: <amount> 10100 <name> Jane Murray <addressLine1> 7 Main St. <city> Amesbury <state> MA <zip> 01913 <country> US <type> MC <number> 5112010100000002 <expDate> 0721 <cardValidationNum> 251		Authorization Response: <response> 301 <message> Invalid Account Number <avsResult> 34 <cardValidationResult> N	

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
8	Authorization/Sale/AVS: <amount> 10100 <name> Mark Johnson <addressLine1> 8 Main St. <city> Manchester <state> NH <zip> 03101 <country> US <type> DI <number> 6011010100000002 <expDate> 0821 <cardValidationNum> 184		Authorization Response: <response> 123 <message> Call Discover <avsResult> 34 <cardValidationResult> P	
9	Authorization/Sale/AVS: <amount> 10100 <name> James Miller <addressLine1> 9 Main St. <city> Boston <state> MA <zip> 02134 <country> US <type> AX <number> 375001010000003 <expDate> 0921 <cardValidationNum> 0421		Authorization Response: <response> 303 <message> Pick Up Card <avsResult> 34 <cardValidationResult> P	
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> 0921 <allowPartialAuth> true	40000 VI 4457010140000141 0921 true	 <response> 010 <message> Partially Approved <approvedAmount> 32000	010 Partially Approved 32000

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
11	<amount> <type> <number> <expDate> <allowPartialAuth>	60000 MC 5112010140000004 1121 true	<response> <message> <approvedAmount>	010 Partially Approved 48000
12	<amount> <type> <number> <expDate> <allowPartialAuth>	50000 AX 375001014000009 0421 true	<response> <message> <approvedAmount>	010 Partially Approved 40000
13	<amount> <type> <number> <expDate> <allowPartialAuth>	15000 DI 6011010140000004 0821 true	<response> <message> <approvedAmount>	010 Partially Approved 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 Insights feature set, you may skip these tests.				
14	<amount> <type> <number> <expDate>	10100 VI 4457010200000247 0821	<response> <message> <type> <availableBalance> <reloadable> <prepaidCardType>	000 Approved PREPAID 2000 NO GIFT
15	<amount> <type> <number> <expDate>	10100 MC 5500000254444445 0321	<response> <message> <type> <availableBalance> <reloadable> <prepaidCardType>	000 Approved PREPAID 2000 YES PAYROLL

TABLE 2-1 Authorization Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
16	<amount> <type> <number> <expDate>	10100 MC 5592106621450897 0321	<response> <message> <type> <availableBalance> <reloadable> <prepaidCardType>	000 Approved PREPAID 0 YES PAYROLL
17	<amount> <type> <number> <expDate>	10100 MC 5590409551104142 0321	<response> <message> <type> <availableBalance> <reloadable> <prepaidCardType>	000 Approved PREPAID 6500 YES PAYROLL
18	<amount> <type> <number> <expDate>	10100 MC 5587755665222179 0321	<response> <message> <type> <availableBalance> <reloadable> <prepaidCardType>	000 Approved PREPAID 12200 YES PAYROLL
19	<amount> <type> <number> <expDate>	10100 MC 5445840176552850 0321	<response> <message> <type> <availableBalance> <reloadable> <prepaidCardType>	000 Approved PREPAID 20000 YES PAYROLL
20	<amount> <type> <number> <expDate>	10100 MC 5390016478904678 0321	<response> <message> <type> <availableBalance> <reloadable> <prepaidCardType>	000 Approved PREPAID 10050 YES 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 Insights feature set, you may skip these tests.				
21	<amount> <type> <number> <expDate>	10100 VI 4100200300012009 0921	<response> <message> <affluence>	000 Approved AFFLUENT
22	<amount> <type> <number> <expDate>	10100 VI 4100200300013007 1121	<response> <message> <affluence>	000 Approved MASS AFFLUENT
23	<amount> <type> <number> <expDate>	10100 MC 5112010201000109 0421	<response> <message> <affluence>	000 Approved AFFLUENT
24	<amount> <type> <number> <expDate>	10100 MC 5112010202000108 0821	<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 Insights feature set, you may skip these tests.				
25	<amount> <type> <number> <expDate>	10100 VI 4100200310000002 1121	<response> <message> <issuerCountry>	000 Approved BRA

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 117.

To test Authorization Reversal Transactions:

1. Verify that your Authorization Reversal XML templates are coded correctly (refer to [Authorization Reversal Transactions](#) on page 220).
2. Submit the Authorizations, Captures (if applicable), and Authorization Reversal Transactions using the orders shown in [Table 2-2](#).
3. Verify that your response values match those shown in Key Response Elements as shown in [Table 2-2](#).

TABLE 2-2 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> 0121 <cardValidationNum> 349		Authorization Response: <response> 000 <message> Approved <authCode> 11111 <avsResult> 01 <cardValidationResult> M	
32A	Capture: <amount> 5050 <littleTxnId> Value returned in Auth response for Order Id 32		Capture Response: <response> 000 <message> Approved	

TABLE 2-2 Authorization Reversal Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
32B	Authorization Reversal: <littleTxnId> <amount>	Value returned in Auth response for Order Id 32 Do Not Submit an Amount	Auth Reversal Response: <response> <message> Declined Transaction report result = 111 - 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.	<response> 000 Approved
33	Authorization: <amount> <name> <addressLine1> <addressLine2> <city> <state> <zip> <country> <type> <number> <expDate> <cardValidationNum> <authenticationValue>	20020 Mike J. Hammer 2 Main St. Apt. 222 Riverside RI 02915 US MC 5112010000000003 0221 261 BwABBJQ1AgAAA AAgJDUCAAAAAA A=	Authorization Response: <response> <message> <authCode> <avsResult> <cardValidationResult> <authenticationResult> Note: Not returned for MasterCard	<response> 000 Approved 22222 10 M
33A	Authorization Reversal: <littleTxnId> <amount>	Value returned in Auth response for Order Id 33 Do Not Submit an amount	Auth Reversal Response: <response> <message>	000 Approved

TABLE 2-2 Authorization Reversal Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
34	Authorization: <amount> 30030 <name> Eileen Jones <addressLine1> 3 Main St. <city> Bloomfield <state> CT <zip> 06002 <country> US <type> DI <number> 6011010000000003 <expDate> 0321 <cardValidationNum> 758		Authorization Response: <response> 000 <message> Approved <authCode> 33333 <avsResult> 10 <cardValidationResult> M	
34A	Authorization Reversal: <littleTxnId> Value returned in Auth response for Order Id 34 <amount> Do Not Submit an Amount		Auth Reversal Response: <response> 000 <message> Approved	
35	Authorization: <amount> 10100 <name> Bob Black <addressLine1> 4 Main St. <city> Laurel <state> MD <zip> 20708 <country> US <type> AX <number> 375001000000005 <expDate> 0421		Authorization Response: <response> 000 <message> Approved <authCode> 44444 <avsResult> 13	

TABLE 2-2 Authorization Reversal Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
35A	Capture: <amount> 5050 <littleTxnId> Value returned in Auth response for Order Id 35		Capture Response: <response> 000 <message> Approved	
35B	Authorization Reversal: <littleTxnId> Value returned in Auth response for Order Id 35 <amount> 5050		Auth Reversal Response: <response> 000 <message> Declined Transaction report result = 336 - Reversal amount does not match Authorization amount	
36	Authorization: <amount> 20500 <type> AX <number> 375000026600004 <expDate> 0521		Authorization Response: <response> 000 <message> Approved	
36A	Authorization Reversal: <littleTxnId> Value returned in Auth response for Order Id 36 <amount> 10000		Auth Reversal Response: <response> 000 <message> Declined Transaction report result = 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 277.)
2. Submit the eCheck Verification transactions using the data sets supplied in [Table 2-3](#).

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, phone, 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 2-3](#).
4. After you complete this test, go to the next eCheck test.

TABLE 2-3 eCheck Verification Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
37	<amount>	3001	<response>	301
	<orderSource>	telephone	<message>	Invalid Account Number
	<firstName>	Tom		
	<lastName>	Black		
	<accType>	Checking		
	<accNum>	10@BC99999		
	<routingNum>	053100300		
38	<amount>	3002	<response>	000
	<orderSource>	telephone	<message>	Approved
	<firstName>	John		
	<lastName>	Smith		
	<accType>	Checking		
	<accNum>	10999999999		
	<routingNum>	011075150		

TABLE 2-3 eCheck Verification Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
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 011075150	<response> <message>	951 Absolute Decline

To test **eCheck Prenotification** transactions:

NOTE: The eCheck Prenotification tests are required only if you plan to perform eCheck Prenotification.

You can only submit eCheck Prenotification transactions in Batches.

1. Verify that your eCheck XML templates are coded correctly (refer to [eCheck Prenotification Credit Transactions \(Batch Only\)](#) on page 266 and [eCheck Prenotification Sale Transactions \(Batch Only\)](#) on page 269.)
2. Submit the eCheck Verification transactions using the data sets supplied in [Table 2-4](#).
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 Prenotification Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
ECPreNoteSale	<orderSource> <name> <accType> <accNum> <routingNum>	ecommerce PreNote Sale Corp Corporate 1092969901 011075150	<response> <message>	000 Approved
ECPreNoteCredit	<orderSource> <name> <accType> <accNum> <routingNum>	ecommerce PreNote Credit Corp Corporate 1099339999 011075150	<response> <message>	000 Approved
PreNoteSaleAccNumErr	<orderSource> <name> <accType> <accNum> <routingNum>	ecommerce PreNote Sale Corp Corporate 10@2969901 011100012	<response> <message>	301 Invalid Account Number
PreNoteCreditAccNumErr	<orderSource> <name> <accType> <accNum> <routingNum>	ecommerce PreNote Credit Personal Savings 10@2969901 011100012	<response> <message>	301 Invalid Account Number
PreNoteSaleRoutNumErr	<orderSource> <name> <accType> <accNum> <routingNum>	ecommerce PreNote Sale Personal Checking 6099999992 053133052	<response> <message>	900 Invalid Bank Routing Number

TABLE 2-4 eCheck Prenotification Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
PreNoteCredit RoutNumErr	<orderSource> <name> <accType> <accNum> <routingNum>	ecommerce PreNote Credit Personal Checking 6099999992 053133052	<response> <message>	900 Invalid Bank Routing Number

To test **eCheck Sale** transactions:

1. Verify that your eCheck XML template is coded correctly (refer to [eCheck Sale Transactions](#) on page 274).
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`, `phone`, 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	<amount> <orderSource> <firstName> <middleInitial> <lastName> <accType> <accNum> <routingNum>	2008 telephone Mike J Hammer Checking 10@BC99999 053100300	<response> <message>	301 Invalid Account Number

TABLE 2-5 eCheck Sale Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
42	<amount> <orderSource> <firstName> <lastName> <accType> <accNum> <routingNum>	2004 telephone Tom Black Checking 4099999992 011075150	<response> <message>	000 Approved
43	<amount> <orderSource> <firstName> <lastName> <companyName> <accType> <accNum> <routingNum>	2007 telephone Peter Green Green Co Corporate 6099999992 011075150	<response> <message>	000 Approved Note: This response will include the accountUpdater element (see accountUpdater on page 345).
44	<amount> <orderSource> <firstName> <lastName> <companyName> <accType> <accNum> <routingNum>	2009 telephone Peter Green Green Co Corporate 9099999992 053133052	<response> <message>	900 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 263.)

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, phone, 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 011075150	<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 345.)
48	<littleTxnId>	Value returned in eCheck Sale response for Order Id 43	<response> <message>	000 Approved

TABLE 2-6 eCheck Credit Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
49	<littleTxnId>	2	<response> <message>	000 Approved

Declined Transaction report result = 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 281.)
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 #46, 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 Vault feature.

NOTE: 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 LittleXML request.

To test explicit Token Registration transactions:

1. Verify that your LittleXML template is coded correctly for this transaction type (refer to [registerTokenRequest](#) on page 693.)
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	<littleToken>	xxxxxxxxxxxx0123
			<bin>	445711
			<type>	VI
			<response>	801
			<message>	Account number was successfully registered
				Note:
51	<accountNumber>	4457119999999999	<littleToken>	none returned
			<response>	820
			<message>	Credit card number was invalid

TABLE 2-7 Register Token Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
52	<accountNumber>	4457119922390123	<littleToken> <bin> <type> <response> <message>	xxxxxxxxxxxx0123 445711 VI 802 Account number was previously registered
53	<accNum> <routingNum>	1099999998 011100012	<littleToken> <type> <eCheckAccountSuffix> <response> <message>	xxxxxxxxxxxx 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 208.)
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, 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 274 and [eCheck Credit Transactions](#) on page 263).
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	<amount> <type> <number> <expDate> <cardValidationNum>	15000 MC 5435101234510196 1121 987	<response> <message> <littleToken> <tokenResponseCode> <tokenMessage> <type> <bin>	000 Approved xxxxxxxxxx0196 801 Account number was successfully registered MC 543510
56	<amount> <type> <number> <expDate> <cardValidationNum>	15000 MC 5435109999999999 1121 987	<<response> <message>	301 Invalid account number
57	<amount> <type> <number> <expDate> <cardValidationNum>	15000 MC 5435101234510196 1121 987	<response> <message> <littleToken> <tokenResponseCode> <tokenMessage> <type> <bin>	000 Approved xxxxxxxxxx0196 802 Account number was previously registered MC 543510

TABLE 2-8 Implicit Registration Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
58	<amount> <littleToken> <expDate> <cardValidationNum>	15000 xxxxxxxxxxxx0196 1121 987 Note: Use the token returned in Order Id 57.	<response> <message>	000 Approved
59	<amount> <littleToken> <expDate>	15000 1111000100092332 1121	<response> <message>	822 Token was not found
60	<amount> <littleToken> <expDate>	15000 1112000100000085 1121	<response> <message>	823 Token was invalid
61	eCheck Sale: <accType> <accNum> <routingNum>	Checking 1099999003 011100012	<littleToken> <tokenResponseCode> <tokenMessage> <type> <eCheckAccountSuffix>	xxxxxxxxxxxxxxxxxxxx 801 Account number was successfully registered EC 003
62	eCheck Sale: <accType> <accNum> <routingNum>	Checking 1099999999 011100012	<littleToken> <tokenResponseCode> <tokenMessage> <type> <eCheckAccountSuffix>	xxxxxxxxxxxxxxxxxxxx 801 Account number was successfully registered EC 999

TABLE 2-8 Implicit Registration Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
63	eCheck Credit: <accType> Checking <accNum> 1099999999 <routingNum> 011100012		<littleToken> <tokenResponseCode> <tokenMessage> <type> <eCheckAccountSuffix>	xxxxxxxxxxxxxxxxxx 801 Account number was successfully registered EC 999
64	eCheck Sale: <accType> Corporate <accNum> 6099999993 <routingNum> 211370545		<originalTokenInfo> <accType> Checking <littleToken> 11190000001003001 <routingNum> 211370545 <newTokenInfo> <accType> Checking <littleToken> 11190000001154101 <routingNum> 211370545 <littleToken> xxxxxxxxxxxxxxxxxx <tokenResponseCode> <tokenMessage> <type> <eCheckAccountSuffix>	(parent element) Checking 11190000001003001 (parent element) Checking 11190000001154101 211370545 xxxxxxxxxxxxxxxxxx 801 Account number was successfully registered EC 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.

2.4.5 Testing Query Transactions

You can use the following test scenarios to verify your coding of the queryTransaction, as well as various responses. To test the query transaction, please do the following:

NOTE: If you are enabled for the use of the queryTransaction, you will not be able to test for response code 153 - Query Transaction not enabled.

1. Submit a queryTransaction with the following elements:
 - <origId>newTransaction</origId> (**Note:** You can use any value never used as an id attribute.)
 - <origActionType>A</origActionType> (**Note:** You can use any valid action type as detailed in the enumerations table of origActionType on page 625.)

Verify that you receive a response value of **151** and message of **Original transaction not found**.
2. Submit an authorization or sale transaction using a unique id attribute.
3. Submit a queryTransaction using the id attribute value from Step 2 as the value for the <origId> element and the <origActionType> element set to the transaction type you submitted in Step 2.

Verify that you receive a response value of **150** and message of **Original transaction found**, a <matchCount> value of 1, and the response message for the transaction submitted in Step 2, as a child of the <result_Max10> element.
4. Submit a second authorization or sale transaction (use the same transaction type as Step 2), using the same value for the id attribute. **Note:** If you use a sale transaction for this test, you must change the credit card number from the one you used in Step 2 to avoid having the transaction flagged as a duplicate.
5. Submit a queryTransaction using the id attribute value from Step 2 as the value for the <origId> element and the <origActionType> element set to the transaction type you submitted in Step 2.

Verify that you receive a response value of **150** and message of **Original transaction found**, a <matchCount> value of 2, and the response messages for the transactions submitted in Steps 2 and 4, as children of the <result_Max10> element.
6. Submit an authorization or sale transaction using a value of **error_id** for the id attribute.
7. Submit a queryTransaction using **error_id** for the origId value.

Verify that you receive a response value of **152** and message of **Original transaction found but response not yet available**, a <matchCount> value of 1, and the <queryTransactionUnavailableResponse> element as a child of the <result_Max10> element.

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 Litle & 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 Security Code No-Match Filtering
- Testing Advanced Fraud Tools
- Testing Account Updater
- Testing Tax Billing
- Testing Convenience Fees
- Testing the Recycling Engine
- Testing Recurring Engine Transactions
- Testing Gift Card Transactions
- Testing Mobile Point of Sale
- Testing MasterPass Transactions
- Testing Apple Pay Transaction Processing
- Testing Android Pay Transaction Processing
- Testing SEPA Direct Debit Transaction
- Testing iDEAL Transactions
- Testing Online Duplicate Transaction Processing
- 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.
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 837](#).

For a list of all possible card validation response codes, see [Card Validation Response Codes on page 841](#).

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

TABLE 2-9 AVS and Card Validation Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
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
76	<type> <number>	MC 5112010400000009	<avsResult> <cardValidationResult>	32 S
78	<type> <number>	MC 5112000600000008	<avsResult> <cardValidationResult>	34 P
80	<type> <number>	AX 374313304211118 Note: American Express CID failures are declined by American Express.	<cardValidationResult> <response> <message>	N 352 Decline CVV2/CID Fail

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 Authorization or Sale transactions in [Table 2-10](#). The AVS tests always return an `avsResult` of 00 when submitting 95 Main St. and 95022. If you extend the Zip Code to 9 digits, by appending 1111, the system returns an `avsResult` of 01, as shown in the second test.

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 as shown in the tests for orderIds AVS3 and AVS4.

For a detailed list of all possible AVS response codes, see [AVS Response Codes](#) on page 837.

TABLE 2-10 AVS and Card Validation Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
AVS1	<type> <number> <addressLine1> <zip>	VI 4457000600000004 95 Main St. 95022	<avsResult>	00
AVS2	<type> <number> <addressLine1> <zip>	VI 4457000600000004 95 Main St. 950221111	<avsResult>	01
AVS3	<type> <number> <addressLine1> <zip>	VI 4457000600000004 100 Main St. 95022	<avsResult>	10
AVS4	<type> <number> <addressLine1> <zip>	VI 4457000600000004 100 Main St. 02134	<avsResult>	20

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 838 for other result codes). To obtain a value of 3 in any position, use one of the following values in the appropriate position:

- <name>Jane Doe</name>
- <phone>5555551234</phone>
- <email>badtest@test.com</email>

For example, if you submit a second transaction using the name Jane Doe Instead of John Doe, the AAVS result would be 311 indicating No Match for name, but Match for phone and email.

TABLE 2-11 Advanced AVS Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
81	<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 1121	<advancedAVSResults>	111

2.5.4 Testing Response Reason Codes and Messages

Use the data provided 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](#). In each case use the supplied card number with a prefix of **RRC-** as the orderId.
2. Verify that you handle the response correctly.

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 812.
- For a list of all possible AVS response codes, see [AVS Response Codes](#) on page 837.

TABLE 2-12 Response Code Test Data

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

TABLE 2-12 Response Code Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
RRC-card #	<type>	AX	<response>	120
	<number>	375000030000001	<message>	Call Issuer
	<type>	DI	<response>	123
	<number>	6011000400000000	<message>	Cal Discover
	<type>	VI	<response>	120
	<number>	4457001200000006	<message>	Call Issuer
	<type>	VI	<response>	120
	<number>	4457001300000005	<message>	Call Issuer
	<type>	VI	<response>	120
	<number>	4457001400000004	<message>	Call Issuer
	<type>	MC	<response>	101
	<number>	5112001000000002	<message>	Issuer Unavailable
	<type>	VI	<response>	321
	<number>	4457001900000009	<message>	Invalid Merchant
	<type>	VI	<response>	303
	<number>	4457002000000006	<message>	Pick Up Card
	<type>	VI	<response>	110
	<number>	4457002100000005	<message>	Insufficient Funds
	<type>	VI	<response>	120
	<number>	4457002200000004	<message>	Call Issuer
	<type>	AX	<response>	350
	<number>	3750000500000006	<message>	Generic Decline
	<type>	VI	<response>	349
	<number>	4457002300000003	<message>	Do Not Honor
	<type>	VI	<response>	340
	<number>	4457002500000001	<message>	Invalid Amount
	<type>	MC	<response>	301
	<number>	5112001600000006	<message>	Invalid Account Number

TABLE 2-12 Response Code Test Data

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

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 tests except the last three Discover tests, set the `orderSource` element to either `3dsAuthenticated` or `3dsAttempted` and the `cardholderAuthentication` element to the following base64 encoded string:

```
BwABBJQ1AgAAAAAgJDUAAAAAAA=
```

The response from a 3DS test will be the same as an Authorization or Sale response, except the `authenticationResult` element will be included in the response as a child of the `fraudResult` element.

TABLE 2-13 3DS Test Data

<code>orderId</code>	Supplied Data Elements		Key Response Elements	
	<code>Element</code>	<code>Value</code>	<code>Element</code>	<code>Value</code>
3DS1	<type>	VI	<authenticationResult>	0
	<number>	4100200300000004		
3DS2	<type>	VI	<authenticationResult>	1
	<number>	4100200300000012		
3DS3	<type>	VI	<authenticationResult>	2
	<number>	4100200300000103		
3DS4	<type>	VI	<authenticationResult>	A
	<number>	4100200300001002		
3DS5	<type>	VI	<authenticationResult>	3
	<number>	4100200300000020		
3DS6	<type>	VI	<authenticationResult>	4
	<number>	4100200300000038		
3DS7	<type>	VI	<authenticationResult>	5
	<number>	4100200300000046		
3DS8	<type>	VI	<authenticationResult>	6
	<number>	4100200300000053		

TABLE 2-13 3DS Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
3DS9	<type> <number>	VI 4100200300000061	<authenticationResult>	7
3DS10	<type> <number>	VI 4100200300000079	<authenticationResult>	8
3DS11	<type> <number>	VI 4100200300000087	<authenticationResult>	9
3DS12	<type> <number>	VI 4100200300000095	<authenticationResult>	B
3DS13	<type> <number>	VI 4100200300000111	<authenticationResult>	C
3DS14	<type> <number>	VI 4100200300000129	<authenticationResult>	D
3DS15	<orderSource> <type> <number>	3dsAttempted MC 5112010200000001	<authenticationResult>	N/A
3DS16	<orderSource> <type> <number>	3dsAttempted MC 5112010200000001	<authenticationResult>	N/A
DI3DS1	<orderSource> <type> <number> <cardholderAuthentication>	3dsAuthenticated DI 6011000400001008 BRIBAIASNBERERER ERERARERERE=	<authenticationResult>	0
DI3DS2	<orderSource> <type> <number> <cardholderAuthentication>	3dsAuthenticated DI 6011000400000018 BRIBAIASNBERERER ERERARERERE=	<authenticationResult>	1

TABLE 2-13 3DS Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
DI3DS3	<orderSource> <type> <number> <cardholderAuthentication>	3dsAuthenticated DI 6011000400000109 BRIAAIASNBERERER ERERARERERE=	<authenticationResult>	2

2.5.6 Testing the Prepaid Filtering Feature

Complete this test only if you are planning on using the 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
filterPrepaidMC	<amount>	9100	<response>	309
	<orderSource>	recurring	<message>	Restricted Card -
	<name>	John Doe		Prepaid Card
	<addressLine1>	10 Main St.		Filtering Service
	<city>	San Jose	<avsResult>	02
	<state>	CA		
	<zip>	95032		
	<country>	US		
	<email>	jdoe@phoenixProcessing.com		
	<phone>	7812701111		
	<type>	MC		
	<number>	5500000958501839		
	<expDate>	1121		
	<prepaid>	true		
filterPrepaidVI	<amount>	9100	<response>	309
	<orderSource>	recurring	<message>	Restricted Card -
	<name>	John Doe		Prepaid Card
	<addressLine1>	10 Main St.		Filtering Service
	<city>	San Jose	<authCode>	none
	<state>	CA	<avsResult>	34
	<zip>	95032		
	<country>	US		
	<email>	jdoe@phoenixProcessing.com		
	<phone>	7812701111		
	<type>	VI		
	<number>	4650002010001478		
	<expDate>	1121		
	<prepaid>	true		

2.5.7 Testing the International Card Filter Feature

Complete this test only if you are planning on using the 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>	1121		

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>	1121		
	<international>	false		

2.5.8 Testing Security Code No-Match Filtering

Complete this test only if you are planning on using the Security Code No-Match Filtering Feature.

To test the Security Code No-Match feature:

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](#).

After you complete this test, go to the next test.

TABLE 2-16 Security Code No-Match Filtering Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
securityCodeFilte r1	<amount> <orderSource> <name> <addressLine1> <city> <state> <zip> <country> <type> <number> <expDate> <cardValidationNum>	9100 ecommerce John Doe 10 Main St. San Jose CA 95032 US MC 5112002200000008 1121 123	<response> <message> <avsResult> <cardValidationRes ult>	358 Restricted by Little due to security code mismatch. 14 N
securityCodeFilte r2	<amount> <orderSource> <name> <addressLine1> <city> <state> <zip> <country> <type> <number> <expDate> <cardValidationNum> <international>	9100 ecommerce Jane Doe 10 Main St. San Jose CA 95032 US MC 5112000200000002 1121 123 false	<response> <message> <avsResult> <cardValidationRes ult>	358 Restricted by Little due to security code mismatch. 20 N

2.5.9 Testing Advanced Fraud Tools

Complete this test only if you are planning on using the Advanced Fraud Tools Feature.

To test the Advanced Fraud Tools feature:

1. Submit authorization transactions using the values provided in Supplied Data Elements of [Table 2-17](#). For each test, replace "Your Prefix-" with the prefix supplied by your Implementation Consultant.
2. Verify that your response values match those shown in Key Response Elements section of [Table 2-17](#).

NOTE: You can submit these tests as sale transactions using the supplied data, or as standalone fraudCheck transactions using just the designated orderId and threatMetrixsessionId. The results for each test type will be as shown in the Key response elements section.

Also, please note that the third test, orderId = tmx_fail_order_id, has two possible results depending upon whether you are configured for Info Only or Auto-Decline.

After you complete this test, go to the next test.

TABLE 2-17 Advanced Fraud Tools Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
tmx_pass_order_id	<amount> <type> <number> <expDate> <threatMetrixSessionId>	150 VI 4111111111111111 1230 Your Prefix-A980A93LP2 O3-KNP0050	<response> <message> <deviceReviewStatus> <deviceReputationScore> <triggeredRule>	000 Approved pass 50 FlashDisabled

TABLE 2-17 Advanced Fraud Tools Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
tmx_review_order_id	<amount> <type> <number> <expDate> <threatMetrixSessionId>	150 VI 4111111111111111 1230 Your Prefix -A0S9D8F7G6H5J4-KMR-020	Result if Info Only: <response> <message> <deviceReviewStatus> <deviceReputationScore> <triggeredRule> <triggeredRule>	000 Approved review -20 PossibleVPNConnection PossibleCookieWipingWeek
tmx_fail_order_id	<amount> <type> <number> <expDate> <threatMetrixSessionId>	150 VI 4111111111111111 1230 Your Prefix -Q1W2E3R4T5Y6U7I8-KHF-100	Result if Info Only: <response> <message> <deviceReviewStatus> <deviceReputationScore> <>triggeredRule> <triggeredRule> <triggeredRule> Result if Auto-decline: <response> <message> <deviceReviewStatus> <deviceReputationScore> <triggeredRule> <triggeredRule> <triggeredRule>	000 Approved fail -100 5PaymentsOnExactDevice ProxyHasNegativeReputation TrueIPProxyIPCityMismatch 550 Restricted Device or IP - ThreatMetrix Fraud Score Below Threshold fail -100 5PaymentsOnExactDevice ProxyHasNegativeReputation TrueIPProxyIPCityMismatch

TABLE 2-17 Advanced Fraud Tools Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
tmx_unavail_order_id	<amount> <type> <number> <expDate> <threatMetrixSessionId>	150 VI 4111111111111111 1230 Your Prefix-Q1W2E3R4T5Y6U7I8-XLP0050	<response> <message> <deviceReviewStatus>	000 Approved unavailable

2.5.10 Testing Account Updater

To test 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 Account Updater service:

1. Submit authorization transactions using the values provided in Supplied Data Elements of [Table 2-18](#).
2. Verify that your response values match those shown in Key Response Elements section of [Table 2-18](#).
3. If you have coded to receive Extended Response Codes, proceed to the next test.

TABLE 2-18 Account Updater Test Data

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

2.5.10.1 Testing Account Updater Extended Response Codes

To test the Account Updater Extended Response Codes feature:

NOTE: You are required to code to LittleXML 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 2-19](#).
2. Verify that the response values match those shown in Key Response Elements section of [Table 2-19](#) and that your systems parse the data correctly. The second test case does not include account repair information only the Extended Response Code.

TABLE 2-19 Account Updater Extended Response Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
102	<amount> <type> <number> <expDate>	10000 MC 5112000101110009 1199	<originalCardInfo> <type> <number> <expDate> <newCardInfo> <type> <number> <expDate> <extendedCardResponse> <code> <message>	(parent element) MC 5112000101110009 1199 (parent element) VI 4457000302200001 1199 (parent element) 501 The account was closed.
103	<amount> <type> <number> <expDate>	10000 VI 4457000301100004 1199	<extendedCardResponse> <code> <message>	(parent element) 504 Contact the cardholder for updated information.

2.5.10.2 Testing Account Updater for Tokenized Merchants

If you are a tokenized merchant using the Account Updater service, you can test this service using the card information provided in [Table 2-18](#). 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 347).

2.5.11 Testing Tax Billing

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 2-20](#). Note: The second transaction omits the <taxType> element.
2. Verify that the system returns a response code of 000 - Approved for the first transaction and response code of 851 - MCC 9311 requires taxType element for the second.

TABLE 2-20 Tax Billing Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
MCC9311Test	<amount> <type> <number> <expDate> <taxType>	3000 VI 4457010200000247 1121 fee	<response> <message>	000 Approved
MCC9311Test2	<amount> <type> <number> <expDate>	3000 VI 4457010200000247 1121	<response> <message>	851 MCC 9311 requires taxType element

2.5.12 Testing Convenience Fees

You include Convenience Fees through the use of the `secondaryAmount` element.

To test the use of Convenience Fees submit the transactions detailed below:

1. Submit authorization or sale transactions for the first five transactions of [Table 2-21](#) using the values provided in Supplied Data Elements columns.
2. Verify that the response values match those shown in Key Response Elements section of [Table 2-21](#) for those transactions and that your systems parse the data correctly.
3. Submit sale transactions for Order Id SaleWOSecondary using the data provided.
4. After receiving an approval for the sale transaction, submit a credit transaction using the `littleTxnId` from the sale transaction and including the `secondaryAmount` element with the value provided.
5. Verify that the response values match those shown in Key Response Elements section for the credit transaction using Order Id SaleWOSecondary and that your systems parse the data correctly.

TABLE 2-21 Convenience Fee Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
Visa_secondaryAmount	<amount> <type> <number> <expDate> <secondaryAmount>	10500 VI 4457010200000247 1121 10000	<response> <message>	000 Approved
SecondaryAmt_Higher	<amount> <type> <number> <expDate> <secondaryAmount>	2500 VI 4111111111111111 1230 3000	<response> <message>	380 Secondary Amount cannot exceed the sale amount
MOP_Unsupported	<amount> <type> <number> <expDate> <secondaryAmount>	6002 AX 37500101000003 0421 1100	<response> <message>	381 This method of payment does not support secondary amount
Negative_Secondary	<amount> <type> <number> <expDate> <secondaryAmount>	1000 VI 4457010200000247 1121 -500	<response> <message>	382 Secondary Amount cannot be less than zero
Partial_Not_Allowed	<amount> <type> <number> <expDate> <secondaryAmount> <allowPartialAuth>	2500 VI 4111111111111111 1230 3000 true	<response> <message>	383 Partial transaction is not supported when including a secondary amount
SaleWOSecondary (Sale TXN)	<amount> <type> <number> <expDate>	1000 MC 5112010140000004 1121	<response> <message>	000 Approved

TABLE 2-21 Convenience Fee Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
SaleWOSecondary (Credit Txn)	<littleTxnId> <amount> <secondaryAmount>	Value from previous transaction 1000 400	<response> <message> Declined Transaction report result = 385 - Secondary amount not allowed on refund if not included on deposit	000 Approved

2.5.13 Testing the Recycling Engine

The Certification test cases for the 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.

There are three test scenarios you can use to test the Recycling Engine and your ability to parse the response messages and/or result Batches. The particular data sets and scenarios you use depends upon the version of LittleXML you use, as well as your plans for retrieving the response messages. Use the following to determine which tests you should run:

- If you plan to retrieve recycling results via the results Batches posted daily to the FTP site, perform the tests in Scenario 1.
- If you are using LittleXML schema version V8.5 or below, perform the tests in Scenario 2.
- If you are using LittleXML schema version V8.6 or above, perform the tests in Scenario 3.

Scenario 1

To test your handling of the Recycling Results Session file:

1. Submit authorization (or sale) transactions using the values provided in the Supplied Data Elements column of [Table 2-22](#). Please use the same value for the orderId and if applicable, the recycleId elements.

NOTE: If your configuration is set for Vantiv to recycle by default, you can omit the <recycleBy> element.

2. Wait a minimum of 2 hours after submitting the last transaction. After 2 hours, retrieve the Results Session file from the FTP site.

TABLE 2-22 Recycling Engine Test Data - Results Session File Pick-up

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase1Order1	<amount> <type> <number> <expDate> <recycleBy>	10000 VI 44570124000000027 1220 Little	Initial Response: <response> 110 <message> Insufficient Funds Final Response (in FTP Session File): <response> 000 <message> Approved	true
XXXCase1Order2	<amount> <type> <number> <expDate> <recycleBy>	10000 MC 5160124000000029 1220 Little	Initial Response: <response> 110 <message> Insufficient Funds Final Response (in FTP Session File): <response> 000 <message> Approved	true
XXXCase1Order3	<amount> <type> <number> <expDate> <recycleBy>	10000 VI 4100200700000059 1220 Little	Initial Response: <response> 110 <message> Insufficient Funds Final Response (in FTP Session File): <response> 110 <message> Insufficient Funds	true

TABLE 2-22 Recycling Engine Test Data - Results Session File Pick-up

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase1Order4	<amount> <type> <number> <expDate> <recycleBy>	10000 MC 5500010000000052 1220 Little	Initial Response: <response> 110 <message> Insufficient Funds Final Response (in FTP Session File): <response> 110 <message> Insufficient Funds	
XXXCase1Order5	<amount> <type> <number> <expDate> <recycleBy>	10000 VI 4457032800000047 1220 Little	Initial Response: <response> 110 <message> Insufficient Funds Final Response (in FTP Session File): <response> 328 <message> Cardholder requests that recurring or installment payment be stopped	
XXXCase1Order6	<amount> <type> <number> <expDate> <recycleBy>	10000 MC 5160328000000042 1220 Little	Initial Response: <response> 110 <message> Insufficient Funds Final Response (in FTP Session File): <response> 120 <message> Call Issuer	

TABLE 2-22 Recycling Engine Test Data - Results Session File Pick-up

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase1Order7	<amount> <type> <number> <expDate> <recycleBy>	10000 VI 5160328000000042 1220 Little	Initial Response: <response> <message> <recyclingEngineActive> Final Response (in FTP Session File): None - This type of decline is not recycled.	302 Account Number Does Not Match Payment Type false

Scenario 2

To test your handling of the Recycling Results Session file and Normal Batch/Online response for schema version V8.5 or below:

1. Submit authorization (or sale) transactions using the values provided in the Supplied Data Elements column of [Table 2-23](#). Please use the same value for the orderId and if applicable, the recycleId elements.

NOTE: If your configuration is set for Vantiv to recycle by default, you can omit the <recycleBy> element.

2. Wait a minimum of 2 hours after submitting the last of the initial transactions. After 2 hours, you can retrieve the Results Session file from the FTP site and/or resubmit the transaction. The responses will contain the data shown for the Final Response.

TABLE 2-23 Recycling Engine Test Data - Online or Results Session File Pick-up V8.5 and below

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase2Orde r1	<amount> <type> <number> <expDate> <recycleBy>	20000 VI 4457012400000027 1220 Little	Initial Response: <response> 110 <message> Insufficient Funds <recyclingEngineA ctive> Final Response (Online/Normal Batch, or in FTP Session File): <response> 000 <message> Approved	
XXXCase2Orde r2	<amount> <type> <number> <expDate> <recycleBy>	20000 MC 5160124000000029 1220 Little	Initial Response: <response> 110 <message> Insufficient Funds <recyclingEngineA ctive> Final Response (Online/Normal Batch, or in FTP Session File): <response> 000 <message> Approved	

TABLE 2-23 Recycling Engine Test Data - Online or Results Session File Pick-up V8.5 and below

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase2Order3	<amount> <type> <number> <expDate> <recycleBy>	20000 VI 4100200700000059 1220 Little	Initial Response: Final Response in Online or Normal Session File: Final Response in FTP Session File:	<response> <message> <recyclingEngineActive> <response> <message> <recyclingEngineActive> <response> <message>
XXXCase2Order4	<amount> <type> <number> <expDate> <recycleBy>	20000 MC 5500010000000052 1220 Little	Initial Response: Final Response in Online or Normal Session File: Final Response in FTP Session File:	<response> <message> <recyclingEngineActive> <response> <message> <recyclingEngineActive> <response> <message>

TABLE 2-23 Recycling Engine Test Data - Online or Results Session File Pick-up V8.5 and below

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase2Order5	<amount> <type> <number> <expDate> <recycleBy>	20000 VI 5160328000000042 1220 Little	Initial Response: <response> <message> <recyclingEngineActive> Final Response (in Session File): None - This type of decline is not recycled.	302 Account Number Does Not Match Payment Type false

Scenario 3

To test your handling of the Intercept Response Codes/Messages, as well as the Recycling Results Session file and Normal Batch/Online responses:

1. Submit authorization (or sale) transactions using the values provided in the Supplied Data Elements column of [Table 2-24](#). Please use the same value for the orderId and if applicable, the recycleId elements.

NOTE: If your configuration is set for Vantiv to recycle by default, you can omit the <recycleBy> element.

2. If you are using schema version V8.6 or above, resubmit any of the first four transactions within 36 hours to receive a response message containing the intercept Response Reason Code 372 - Soft Decline - Auto Recycling In Progress. If you are using schema version 8.5 or below, you will receive a response message with the same Response Reason Code as in the initial response message.
3. Wait a minimum of 36 hours after submitting the last of the initial transactions. After 36 hours, you can retrieve the Results Session file from the FTP site and/or resubmit the transaction. The responses will contain the data shown for the Final Response.

TABLE 2-24 Recycling Engine Test Data - Intercept and Online or Results Session File Pick-up

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase3Order1	<amount> <type> <number> <expDate> <recycleBy>	20000 DI 6223012400000025 1220 Little	Initial Response: <response> 350 <message> Generic Decline <recyclingEngineActive> Intermediate Attempts (V8.6): <response> 372 <message> Soft decline - Recycling In Progress <recyclingEngineActive> Final Response (Online/Normal Batch, or in FTP Session File): <response> 000 <message> Approved	true true true

TABLE 2-24 Recycling Engine Test Data - Intercept and Online or Results Session File Pick-up

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase3Order2	<amount> <type> <number> <expDate> <recycleBy>	20000 AX 377201240000025 1220 Little	Initial Response: <response> 350 <message> Generic Decline <recyclingEngineActive> Intermediate Attempts (V8.6): <response> 372 <message> Soft decline - Recycling In Progress <recyclingEngineActive> Final Response (Online/Normal Batch, or in FTP Session File): <response> 000 <message> Approved	true true true

TABLE 2-24 Recycling Engine Test Data - Intercept and Online or Results Session File Pick-up

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase3Order3	<amount> <type> <number> <expDate> <recycleBy>	20000 DI 6223012400000033 1220 Little	Initial Response: <response> 350 <message> Generic Decline Intermediate Attempts (V8.6): <response> 372 <message> Soft decline - Recycling In Progress <recyclingEngineActive> Final Response (Online/Normal Batch, or in FTP Session File): <response> 373 <message> Hard Decline - Auto Recycling Complete	true

TABLE 2-24 Recycling Engine Test Data - Intercept and Online or Results Session File Pick-up

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase3Order4	<amount> <type> <number> <expDate> <recycleBy>	20000 DI 6011002078551608 1220 Little	Initial Response: <response> 101 <message> Issuer Unavailable Intermediate Attempts (V8.6): <response> 372 <message> Soft decline - Recycling In Progress <recyclingEngineActive> Final Response (Online/Normal Batch, or in FTP Session File): <response> 373 <message> Hard Decline - Auto Recycling Complete	true
XXXCase3Order5	<amount> <type> <number> <expDate> <recycleBy>	20000 VI 377203280000048 1220 Little	Initial Response: <response> 302 <message> Account Number Does Not Match Payment Type All Responses: None - This type of decline is not recycled.	false

2.5.13.1 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 a sale transaction using the values provided for Case1Order1a and an authorization transaction using the values provided for Case1Order2a in the Supplied Data Elements of [Table 2-25](#).
2. Two (2) hours after receiving the decline message, submit a void transaction using the littleTxnId returned in the response message for Case1Order1a. If you are not enabled for Auto-refunding an approved Sales on Void, the Declined transaction report will contain a response code of 362 - Transaction not Voided - Already Settled.

NOTE: If you submit the Void transaction (Step 2) within 2 hours of the initial transaction submission, you will receive a voidResponse with a response code of 000 - Approved.

3. After receiving the decline messages, submit an authReversal transaction using the littleTxnId returned in the response message for Case1Order2a. This will halt the recycling of the order.

TABLE 2-25 Recycling Engine Cancellation Test Data

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase1Orde r1a	<amount> 11000 <type> VI <number> 4457012400000027 <expDate> 1220 <recycleBy> Little Submit Void using littleTxnId from Initial Response		Initial Response: <response> 110 <message> Insufficient Funds <recyclingEngineA ctive> Void Response (if enabled for Auto-Refund): <response> 000 <message> <creditLittleTxnId> (Random Value) Void Response (if not enabled for Auto-Refund: <response> <message> Declined Transaction report result = 362 - Transaction Not Voided - Already Settled	true 000 Approved (Random Value) 000 Approved

TABLE 2-25 Recycling Engine Cancellation Test Data

orderId or recycleId (replace XXX with your merchantId)	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
XXXCase1Order2a	<amount> 11000 <type> MC <number> 5160124000000029 <expDate> 1220 <recycleBy> Little Submit authReversal using littleTxnId from Initial Response		Initial Response: <response> 110 <message> Insufficient Funds <recyclingEngineActive> true authReversalResponse: <response> 000 <message> Approved	

2.5.14 Testing Recurring Engine Transactions

Use the following Certification tests to verify transactions associated with the Recurring Engine functionality. For testing purposes, the Certification environment will process the first recurring transaction within 2 hours. The system cancels subsequent recurring transactions in the payment stream. For example, if you submitted an Authorization at 9:00 AM that set-up a subscription for twelve monthly payments, the Cert Recurring Engine would process the first payment by 11:00 AM and cancel the remaining eleven payments.

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.

To test the Recurring Engine functionality using the data supplied in [Table 2-26](#):

1. Submit `createPlan` transactions for Order Ids R1.1, R1.2, R1.3, R1.4, R1.5, and R1.6. These transactions establish Plans used in the remaining tests, as well as verify your LittleXML used to create Plans. You can use whatever values you wish for the `<planCode>`

- and <name> elements. Verify that each transaction receives an approval code in the response message.
2. Submit `createPlan` transactions for Order Id R1.7 using the same <planCode> value you used in R1.3. This transaction is declined, because the Plan Code already exists. Verify that the transaction receives a response code of 487 - Plan code already exists.
 3. Submit a `sale` transaction for Order Id R2.1. This transaction creates a subscription using the plan established in Order R1.1. The amount in the sale transaction represents a set-up fee and not the initial payment. Verify that the transaction receives an approval code in the response message.
 4. Submit an authorization transaction for Order Id R2.2. This transaction creates a subscription using the plan established in Order R1.1. Because the Authorization did not specify a start date, the Recurring Engine schedules the first payment for the current date. Verify that the transaction receives an approval code in the response message.

NOTE:

The transaction specified in Order 2.2 is a \$0 Auth. If you include an amount when using an Auth to establish a subscription, you should plan on reversing the Auth to avoid a Misuse of Auth fee. The Recurring Engine obtains its own Auth for the first payment.

5. Submit an `updateSubscription` transaction for Order Id R2.3. This transaction updates the subscription initiated with Order Id 2.1, changing the Plan to the plan you created in R1.2.
6. Submit a `sale` transaction for Order Id R3.1. This transaction utilizes a Sale transaction to initialize a subscription based upon the plan created in R1.3, but overrides both the number of payments and the amount specified in the Plan. Verify that the transaction receives an approval code in the response message.
7. Submit an authorization transaction for Order Id R3.2. This transaction utilizes an Authorization transaction to initialize a subscription based upon the plan created in R1.3, but overrides both the number of payments and the amount specified in the Plan. Verify that the transaction receives an approval code in the response message.
8. Submit an `updateSubscription` transaction for Order Id R3.3. This transaction updates an existing subscription, changing the billing date.
9. Submit an authorization transaction for Order Id R4.1. This transaction utilizes an Authorization transaction to initialize a subscription that overrides the default amount in the Plan and has a trial period. Verify that the transaction receives an approval code in the response message.
10. Submit an `updateSubscription` transaction for Order Id R4.2. This transaction updates an existing subscription with a new credit card number.
11. Submit an authorization transaction for Order Id R5.1. This transaction utilizes an Authorization transaction to initialize a subscription based upon the SEMIANNUAL_PLAN, but overrides the number of payments. Verify that the transaction receives an approval code in the response message.

12. Submit an `updateSubscription` transaction for Order Id R5.2. This transaction updates an existing subscription with new billing information.
13. Submit an `updateSubscription` transaction for Order Id R5.3. Since the transaction does not include any updated information, the Declined Transaction report will contain a response code of 484 - Insufficient data to update subscription. Verify the response code by accessing the Declined Transaction report in Vantiv iQ.
14. Submit an `updateSubscription` transaction for Order Id R5.4. Since the transaction specifies a new billing date in the past, the Declined Transaction report will contain a response code of 485 - Invalid billing date. Verify the response code by accessing the Declined Transaction report in Vantiv iQ.
15. Submit an `authorization` transaction for Order Id R6.1. The system declines this transaction, since it uses an invalid Plan Code. Verify that the transaction receives a response code of 472 - Invalid plan code.
16. Submit an `updateSubscription` transaction for Order Id R6.2. Since this order uses an invalid subscription Id, The system declines this transaction, the Declined Transaction report will contain a response code of 482 - Invalid start date. Verify the response code by accessing the Declined Transaction report in Vantiv iQ.
17. Submit an `authorization` transaction for Order Id R6.3. The system declines this transaction, since it uses a start date in the past. Verify that the transaction receives a response code of 475 - Invalid Subscription Id.
18. Submit an `authorization` transaction for Order Id R7.1. This transaction utilizes an Authorization transaction to initialize a subscription based upon the `PLAN_WITH_PROMOTIONS` Plan with an add On and a Discount applied. Verify that the transaction receives an approval code in the response message.
19. Submit an `updateSubscription` transaction for Order Id R7.2. This transaction attempts to update an existing subscription with a new Add On, but is declined because the Add On already exists. Verify that the Declined Transaction report contains a response code of 476 - Add On already exists.
20. Submit an `updateSubscription` transaction for Order Id R7.3. This transaction attempts to update an Add On for an existing subscription, but is declined because the specified Add On is not associated with the Subscription. Verify that the Declined Transaction report contains a response code of 478 - No matching Add On code for the subscription.
21. Submit an `updateSubscription` transaction for Order Id R7.4. This transaction attempts to update an Add On for an existing subscription, but is declined because the specified Add On is not associated with the Subscription. Verify that the Declined Transaction report contains a response code of 478 - No matching Add On code for the subscription.
22. Submit an `updateSubscription` transaction for Order Id R7.5. This transaction attempts to update a Discount for an existing subscription, but is declined because the specified Discount is not associated with the Subscription. Verify that the Declined Transaction report contains a response code of 480 - No matching Discount code for the subscription.
23. Submit an `authorization` transaction for Order Id R8.1. This transaction utilizes an Authorization transaction to initialize a subscription, but includes duplicate Add Ons. Verify that the transaction receives a response code of 477 - Duplicate add-on codes in request.

24. Submit an authorization transaction for Order Id R8.2. This transaction utilizes an Authorization transaction to initialize a subscription, but includes duplicate Discounts. Verify that the transaction receives a response code of 481 - Duplicate discount codes in request.
25. Submit an authorization transaction for Order Id R9.1. In this case the parent Authorization transaction is declined with a response code of 301 - Invalid Account Number. The response code associated with the recurring request is 471 - Parent transaction declined - Recurring subscription not created.PREMIUM_MONTHLY
26. Submit an cancelSubscription transaction for Order Id R10.1. Verify that the transaction receives an approval code in the response message.
27. Submit an updatePlan transaction for Order Id R11.1. This transaction changes the Plan to an inactive state. Verify that the transaction receives an approval code in the response message.

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R1.1	<createPlan> <planCode> <name> <description> <intervalType> <amount> <numberOfPayments>	(parent element) Your Value Your Value Basic monthly plan MONTHLY 5000 12	<response> <message> <planCode>	000 Approved Your Value
R1.2	<createPlan> <planCode> <name> <description> <intervalType> <amount> <numberOfPayments>	(parent element) Your Value Your Value Premium monthly plan MONTHLY 7000 12	<response> <message> <planCode>	000 Approved Your Value
R1.3	<createPlan> <planCode> <name> <description> <intervalType> <amount>	(parent element) Your Value Your Value An annual plan ANNUAL 14000	<response> <message> <planCode>	000 Approved Your Value

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R1.4	<createPlan> <planCode> <name> <description> <intervalType> <amount> <trialIntervalType> <trialNumberofIntervals>	(parent element) Your Value Your Value A monthly plan with trial period MONTHLY 5000 MONTH 2	<response> <message> <planCode>	000 Approved Your Value
R1.5	<createPlan> <planCode> <name> <description> <intervalType> <amount>	(parent element) Your Value Your Value A semi-annual plan SEMIANNUAL 7000	<response> <message> <planCode>	000 Approved Your Value
R1.6	<createPlan> <planCode> <name> <description> <intervalType> <amount>	(parent element) Your Value Your Value A plan with Add Ons and Discounts MONTHLY 5000	<response> <message> <planCode>	000 Approved Your Value
R1.7	<createPlan> <planCode> <name> <description> <intervalType> <amount>	(parent element) Value from R1.3 Value from R1.3 An annual plan ANNUAL 5000	<response> <message> <planCode>	487 Plan code already exists Value from R1.3

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R2.1	<sale> <amount> <recurringRequest> <subscription> <planCode> <startDate>	(parent element) 1000 (parent element) (parent element) Value from R1.1 Use any date	<recurringResponse> <responseCode> <responseMessage>	(parent element) 470 Approved - Recurring subscription created
R2.2	<authorization> <amount> <recurringRequest> <subscription> <planCode>	(parent element) 000 (parent element) (parent element) Value from R1.1	<recurringResponse> <responseCode> <responseMessage>	(parent element) 470 Approved - Recurring subscription created
R2.3	<updateSubscription> <subscriptionId> <planCode>	(parent element) Value returned in R2.1 response Value from R1.2	<subscriptionId> <response> <message>	(parent element) 000 Approved
R3.1	<sale> <recurringRequest> <subscription> <planCode> <startDate> <numberOfPayments> <amount>	(parent element) (parent element) (parent element) Value from R1.3 Use any valid date in the future. 6 4000	<recurringResponse> <responseCode> <responseMessage>	(parent element) 470 Approved - Recurring subscription created
R3.2	<authorization> <recurringRequest> <subscription> <planCode> <startDate> <numberOfPayments> <amount>	(parent element) (parent element) (parent element) Value from R1.3 Use any valid date in the future. 6 4000	<recurringResponse> <responseCode> <responseMessage>	(parent element) 470 Approved - Recurring subscription created

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R3.3	<updateSubscription> <subscriptionId> <billingDate>	(parent element) Value returned in R3.1 response Use any valid date in the future and different from date used in R3.1.	<response> <message>	000 Approved
R4.1	<authorization> <recurringRequest> <subscription> <planCode> <startDate> <amount>	(parent element) (parent element) (parent element) Value from R1.4 Use any valid date in the future. 4000	<recurringResponse> <responseCode> <responseMessage>	(parent element) 470 Approved - Recurring subscription created
R4.2	<updateSubscription> <subscriptionId> <card> <type> <number> <expDate>	(parent element) Value returned in R4.1 response (parent element) VI 4457010000000009 1221	<response> <message>	000 Approved
R5.1	<authorization> <recurringRequest> <subscription> <planCode> <startDate> <numberOfPayments>	(parent element) (parent element) (parent element) Value from R1.5 Use any valid date in the future. 6	<recurringResponse> <responseCode> <responseMessage>	(parent element) 470 Approved - Recurring subscription created

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R5.2	<updateSubscription> <subscriptionId> <billToAddress> <name>John <addressLine1>900 Chelmsford St. <city>Lowell <state>MA <zip>01781 <country>US <email>john@gmail.com <phone>8559658965	(parent element) Value returned in R5.1 response (parent element) John 900 Chelmsford St. Lowell MA 01781 US john@gmail.com 8559658965	<subscriptionId> <response> <message>	Value returned in R5.1 response 000 Approved
R5.3	<updateSubscription> <subscriptionId>	(parent element) Value returned in R5.1 response	<response> <message>	000 Approved
			Declined Transaction report result = 484 - Insufficient data to update subscription	
R5.4	<updateSubscription> <subscriptionId> <billingDate>	(parent element) Value returned in R5.1 response 2000-10-01	<response> <message>	000 Approved
			Declined Transaction report result = 485 - Invalid billing date	
R6.1	<authorization> <recurringRequest> <subscription> <planCode>	(parent element) (parent element) (parent element) INVALID_PLAN	<responseCode> <responseMessage>	472 Invalid plan code

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R6.2	<updateSubscription> <subscriptionId> <billToAddress> <name>John <addressLine1>900 Chelmsford St. <city>Lowell <state>MA <zip>01781 <country>US <email>john@gmail.com <phone>8559658965	(parent element) 1111111111 (parent element) John 900 Chelmsford St. Lowell MA 01781 US john@gmail.com 8559658965	<response> <message>	000 Approved Declined Transaction report result = 475 - Invalid Subscription Id
R6.3	<authorization> <recurringRequest> <subscription> <planCode>Value from R1.1 <startDate>2000-04-04	(parent element) (parent element) (parent element) Value from R1.1 2000-04-04	<recurringResponse> <responseCode> <responseMessage>	(parent element) 482 Invalid start date

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R7.1	<authorization> <recurringRequest> <subscription> <planCode> Value from R1.6 <parent element> SSL_ADDON <addOnCode> Additional Service <name>1000 <amount>2013-08-30 <startDate>2050-08-30 <endDate> <createDiscount> <discountCode> Special Offer <name>1000 <amount>2013-08-30 <startDate>2050-08-30 <endDate>	(parent element) (parent element) (parent element) Value from R1.6 (parent element) SSL_ADDON Additional Service 1000 2013-08-30 2050-08-30 (parent element) PROMO_DISCOUNT Special Offer 1000 2013-08-30 2050-08-30	<recurringResponse> <responseCode> <responseMessage>	(parent element) 470 Approved - Recurring Subscription Created
R7.2	<updateSubscription> <subscriptionId> Value returned in R7.1 response <createAddOn> <addOnCode> SSL_ADDON <name>Additional Service <amount>1000 <startDate>2013-08-30 <endDate>2050-08-30	(parent element) (parent element) Value returned in R7.1 response (parent element) SSL_ADDON Additional Service 1000 2013-08-30 2050-08-30	<recurringResponse> <response> <message>	(parent element) 000 Approved Declined Transaction report result = 476 - Add-on code already exists

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R7.3	<updateSubscription> <subscriptionId> <createDiscount> <discountCode> <name> <amount> <startDate> <endDate>	(parent element) Value returned in R7.1 response (parent element) PROMO_DISCOUNT Special Offer 1000 2013-08-30 2050-08-30	<response> <message> Declined Transaction report result = 486 - Discount code already exists	000 Approved
R7.4	<updateSubscription> <subscriptionId> <updateAddOn> <addOnCode> <name> <amount>	(parent element) Value returned in R7.1 response (parent element) INVALID_ADDON_CODE Extra Features 1000	<response> <message> Declined Transaction report result = 478 - No matching Add-on code for the subscription	478 No matching Add-on code for the subscription
R7.5	<updateSubscription> <subscriptionId> <updateDiscount> <discountCode> <name> <amount> <startDate> <endDate>	(parent element) Value returned in R7.1 response (parent element) INVALID_DISCOUNT_CODE Special Offer 1000 2013-08-30 2050-08-30	<response> <message> Declined Transaction report result = 480 - No matching Discount code for the subscription	480 No matching Discount code for the subscription

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R8.1	<authorization> <recurringRequest> <subscription> <planCode> Value from R1.6 <parent element> <createAddOn> SSL_ADDON <addOnCode> Additional Service <name> 1000 <amount> 2013-08-30 <startDate> 2050-08-30 <endDate> <parent element> <createAddOn> SSL_ADDON <addOnCode> Additional Service <name> 1000 <amount> 2013-08-30 <startDate> 2050-08-30 <endDate>	(parent element) (parent element) (parent element) Value from R1.6 (parent element) SSL_ADDON Additional Service 1000 2013-08-30 2050-08-30 (parent element) SSL_ADDON Additional Service 1000 2013-08-30 2050-08-30	<recurringResponse> <response> <message>	(parent element) 477 Duplicate Add-on codes in request

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R8.2	<authorization> <recurringRequest> <subscription> <planCode> Value from R1.6 <parent element> <createDiscount> <discountCode> <name> 1000 <amount> 2013-08-30 <startDate> 2050-08-30 <endDate> <parent element> <createDiscount> <discountCode> <name> 1000 <amount> 2013-08-30 <startDate> 2050-08-30 <endDate>	(parent element) (parent element) (parent element) Value from R1.6 (parent element) PROMO_DISCOUNT Discount 1000 2013-08-30 2050-08-30 (parent element) PROMO_DISCOUNT Discount 1000 2013-08-30 2050-08-30	<recurringResponse> <responseCode> <responseMessage>	(parent element) 481 Duplicate discount codes in request
R9.1	<authorization> <card> <type> MC <number> 5112010100000002 <expDate> 0721 <recurringRequest> <subscription> <planCode> Value from R1.6	(parent element) (parent element) MC 5112010100000002 0721 (parent element) (parent element) Value from R1.6	<recurringResponse> <responseCode> <responseMessage>	(parent element) 471 Parent transaction declined - Recurring subscription not created
R10.1	<cancelSubscription> <subscriptionId>	(parent element) Value returned in R7.1 response	<recurringResponse> <subscriptionId> <response> 000 <message>	(parent element) Submitted value 000 Approved

TABLE 2-26 Recurring Engine Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
R11.1	<updatePlan> <planCode> <active>	(parent element) Value from R1.2 false	<response> <message>	000 Approved

2.5.15 Testing Gift Card Transactions

Use the following Certification tests to verify transactions associated with the Private Label Gift Card functionality. The tests are designed so that you can verify your XML structure for each transaction type, as well as your ability to parse the response data.

To test the Gift Card functionality using the data supplied in [Table 2-27](#):

NOTE: When processing gift cards, many transactions response messages include the refCode, sequenceNumber, systemTraceId, and txnTime elements. You should always store these values for possible use in subsequent transactions. For example, to perform the giftCardCapture in test GC2A, you must include the values returned for these elements in the associated authorization transaction, GC2.

1. Submit an activate transaction for Order Id GC1. Verify that the data in the response message matches the Key Response Elements specified in the table.
2. Submit an activate transaction for Order Id GC1A. This transaction activates a Virtual Gift Card. Verify that the data in the response message matches the Key Response Elements specified in the table.
3. Submit an authorization transaction for Order Id GC2. Verify that the data in the response message matches the Key Response Elements specified in the table.
4. Submit a giftCardCapture transaction for Order Id GC2A. Verify that the data in the response message matches the Key Response Elements specified in the table.
5. Submit a giftCardCredit transaction for Order Id GC2B. Verify that the data in the response message matches the Key Response Elements specified in the table.
6. Submit a deactivate transaction for Order Id GC3. Verify that the data in the response message matches the Key Response Elements specified in the table.
7. Submit a load transaction for Order Id GC4. Verify that the data in the response message matches the Key Response Elements specified in the table.
8. Submit a unload transaction for Order Id GC5. Verify that the data in the response message matches the Key Response Elements specified in the table.

9. Submit a `balanceInquiry` transaction for Order Id GC6. Verify that the data in the response message matches the Key Response Elements specified in the table.
10. Submit an `activate` transaction for Order Id GC7. Verify that the data in the response message matches the Key Response Elements specified in the table.
11. Submit an `activateReversal` transaction for Order Id GC7A. Verify that the data in the response message matches the Key Response Elements specified in the table.
12. Submit an `activateReversal` transaction for Order Id GC1A. Verify that the data in the response message matches the Key Response Elements specified in the table.
13. Submit an `authorization` transaction for Order Id GC8. Verify that the data in the response message matches the Key Response Elements specified in the table.
14. Submit an `authorizationReversal` transaction for Order Id GC8A. Verify that the data in the response message matches the Key Response Elements specified in the table.
15. Submit a `sale` transaction for Order Id GC9. Verify that the data in the response message matches the Key Response Elements specified in the table.
16. Submit a `depositReversal` transaction for Order Id GC9A. Verify that the data in the response message matches the Key Response Elements specified in the table.
17. Submit a `sale` transaction for Order Id GC10. Verify that the data in the response message matches the Key Response Elements specified in the table.
18. Submit a `giftCardCredit` transaction for Order Id GC10A. Verify that the data in the response message matches the Key Response Elements specified in the table.
19. Submit a `refundReversal` transaction for Order Id GC10B. Verify that the data in the response message matches the Key Response Elements specified in the table.
20. Submit a `deactivate` transaction for Order Id GC11. Verify that the data in the response message matches the Key Response Elements specified in the table.
21. Submit a `loadReversal` transaction for Order Id GC12. Verify that the data in the response message matches the Key Response Elements specified in the table.
22. Submit an `unload` transaction for Order Id GC13. Verify that the data in the response message matches the Key Response Elements specified in the table.
23. Submit an `unloadReversal` transaction for Order Id GC13A. Verify that the data in the response message matches the Key Response Elements specified in the table.
24. Submit an `activate` transaction for Order Id GC14. This transaction is declined with a response code of 301 - Invalid Account Number. Verify that the data in the response message matches the Key Response Elements specified in the table.
25. Submit an `activate` transaction for Order Id GC15. This transaction is declined with a response code of 217 - Already active, because it attempts to activate the same card already activated in Order Id GC1. Verify that the data in the response message matches the Key Response Elements specified in the table.
26. Submit an `authorization` transaction for Order Id GC16. This transaction is declined with a response code of 110 - Insufficient Funds. Verify that the data in the response message matches the Key Response Elements specified in the table.

27. Submit an authorization transaction for Order Id GC17. This transaction is declined with a response code of 281 - Card not active. Verify that the data in the response message matches the Key Response Elements specified in the table.
28. Submit a sale transaction for Order Id GC19. Verify that the data in the response message matches the Key Response Elements specified in the table.
29. Submit a giftCardCredit transaction for Order Id GC19A. For this transaction the Declined Transaction report will contain response code of 304 - Lost/Stolen Card.
30. Submit a balanceInquiry transaction for Order Id GC20. This transaction is declined with a response code of 352 - Invalid CVV2. Verify that the data in the response message matches the Key Response Elements specified in the table.

TABLE 2-27 Private Label Gift Card Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
GC1	Activate: <amount> 15000 <type> GC <number> Supplied by Implementation Consultant <expDate> 1220 <cardValidationNum> 123		Activate Response: <response> 000 <message> Approved <cardValidationResult> M <availableBalance> 15000	
GC1A	Virtual Activate: <amount> 8000 <accountNumberLength> 16 <giftCardBin> Supplied by Implementation Consultant		Activate Response: <response> 000 <message> Approved <availableBalance> 8000 <accountNumber> 603571xxxxxxxxxx xx	
GC2	Authorization: <amount> 1500 <type> GC <number> Number from GC1 <expDate> 1220 <cardValidationNum> 123		Authorization Response: <response> 000 <message> Approved <authCode> 11111 <cardValidationResult> M <availableBalance> 13500	

TABLE 2-27 Private Label Gift Card Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
GC2A	Gift Card Capture: <littleTxnId> <captureAmount> <type> <number> <expDate> <cardValidationNum> <originalRefCode> <originalAmount> <originalTxnTime>	Value from GC2 resp. 1500 GC Number from GC2 1220 123 refCode from GC2 amount from GC2 txnTime from GC2	Capture Response: <response> <message>	000 Approved
GC2B	Gift Card Credit: <littleTxnId> <creditAmount> <type> <number> <expDate>	Value from GC2A resp. 500 GC Number from GC2 1220	Credit Response: <response> <message>	000 Approved
CG3	Deactivate: <type> <number>	GC Supplied by Implementation Consultant	Deactivate Response: <response> <message> <availableBalance>	000 Approved 0
GC4	Load: <amount> <type> <number> <expDate> <cardValidationNum>	5000 GC Supplied by Implementation Consultant 0121 123	Load Response: <response> <message> <cardValidationResu lt> <availableBalance>	000 Approved M 5000

TABLE 2-27 Private Label Gift Card Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
GC5	Unload: <amount> 2500 <type> GC <number> Number from GC4 <expDate> 0121 <cardValidationNum> 123		Unload Response: <response> 000 <message> Approved <cardValidationResu lt> M <availableBalance> 2500	
GC6	Balance Inquiry: <type> GC <number> Number from GC1 <expDate> 0121 <cardValidationNum> 123		Balance Inquiry Response: <response> 000 <message> Approved <cardValidationResu lt> M <availableBalance> 2500	
GC7	Activate: <amount> 8000 <type> GC <number> Supplied by Implementation Consultant <expDate> 1215 <cardValidationNum> 123		Activate Response: <response> 000 <message> Approved <cardValidationResu lt> M <availableBalance> 8000	
GC7A	Activate Reversal: <littleTxnId> Value from GC7 resp. <type> from GC7 <number> from GC7 <expDate> from GC7 <originalRefCode> refCode from GC7 <originalAmount> amount from GC7 <originalSystemTraceld> systemTraceld from GC7 <originalSequenceNumb er> sequenceNumber from GC7		Activate Reversal Response: <response> 000 <message> Approved	

TABLE 2-27 Private Label Gift Card Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
GC7B	Activate Reversal: <littleTxnId> Value from GC1A resp. <type> from GC1 <number> from GC1 <expDate> from GC1 <originalRefCode> refCode from GC1A <originalAmount> amount from GC1A <originalSystemTraceld> systemTraceld from GC1A <originalSequenceNumber> sequenceNumber from GC1A		Activate Reversal Response: <response> 000 <message> Approved	
GC8	Authorization: <amount> 2000 <type> GC <number> Supplied by Implementation Consultant <expDate> 1220 <cardValidationNumber> 123		Authorization Response: <response> 000 <message> Approved <authCode> 11111 <cardValidationResult> M <availableBalance> 4000	
GC8A	Gift Card Auth Reversal: <littleTxnId> Value from GC8 resp. <type> from GC8 <number> from GC8 <expDate> from GC8 <originalRefCode> refCode from GC8 <originalAmount> amount from GC8 <originalSystemTraceld> systemTraceld from GC8 <originalSequenceNumber> sequenceNumber from GC8		Auth Reversal Response: <response> 000 <message> Approved	

TABLE 2-27 Private Label Gift Card Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
GC9	Sale: <amount> 2000 <type> GC <number> Supplied by Implementation Consultant <expDate> 1220 <cardValidationNum> 123		Sale Response: <response> 000 <message> Approved <authCode> 11111 <cardValidationResult> M <availableBalance> 8000	
GC9A	Deposit Reversal: <littleTxnId> Value from GC9 resp. <type> from GC9 <number> from GC9 <expDate> from GC9 <originalRefCode> refCode from GC9 <originalAmount> amount from GC9 <originalSystemTraceId> systemTraceId from GC9 <originalSequenceNumber> sequenceNumber from GC9		Deposit Reversal Response: <response> 000 <message> Approved	
GC10	Sale: <amount> 2000 <type> GC <number> Number from GC9 <expDate> 1220 <cardValidationNum> 123		Sale Response: <response> 000 <message> Approved <authCode> 11111 <cardValidationResult> M <availableBalance> 8000	
GC10A	Gift Card Credit: <littleTxnId> Value from GC10 resp. <CreditAmount> 2000 <type> GC <number> Number from GC9 <expDate> 1220		Credit Response: <response> 000 <message> Approved	

TABLE 2-27 Private Label Gift Card Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
GC10B	Refund Reversal: <littleTxnId> Value from GC10A resp. <type> from GC10 <number> form GC10 <expDate> from GC10 <originalRefCode> refCode from GC10A <originalAmount> amount from GC10A <originalSystemTraceld> systemTraceld from GC10A <originalSequenceNumber> sequenceNumber from GC10A		Refund Reversal Response: <response> 000 <message> Approved	
GC11	Deactivate Reversal: <littleTxnId> Value from GC3 resp. <type> from GC3 <number> from GC3 <expDate> from GC3 <originalRefCode> refCode from GC3 resp. <originalAmount> amount from GC3 resp. <originalSystemTraceld> systemTraceld from GC3 <originalSequenceNumber> sequenceNumber from GC3 resp.		Deactivate Reversal Response: <response> 000 <message> Approved	
GC12	Load Reversal: <littleTxnId> Value from GC4 resp. <type> from GC4 <number> from GC4 <expDate> from GC4 <originalRefCode> refCode from GC4 resp. <originalAmount> amount from GC4 resp. <originalSystemTraceld> systemTraceld from GC4 <originalSequenceNumber> sequenceNumber from GC4 resp.		Load Reversal Response: <response> 000 <message> Approved <availableBalance> 0	

TABLE 2-27 Private Label Gift Card Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
GC13	Unload: <amount> 2500 <type> GC <number> Supplied by Implementation Consultant <expDate> 0121 <cardValidationNum> 123		Unload Response: <response> 000 <message> Approved <cardValidationResult> M <availableBalance> 1500	
GC13A	Unload Reversal: <littleTxnId> Value from GC13 resp. <type> from GC13 <number> from GC13 <expDate> from GC13 <originalRefCode> refCode from GC13 resp. <originalAmount> amount from GC13 resp. <originalSystemTraceId> systemTraceId from GC13 resp. <originalSequenceNumber> sequenceNumber from GC13 resp.		Unload Reversal Response: <response> 000 <message> Approved	
GC14	Activate: <amount> 15000 <type> GC <number> Supplied by Implementation Consultant <expDate> 1221 <cardValidationNum> 123		Activate Response: <response> 301 <message> Invalid Account Number	
GC15	Activate: <amount> 10000 <type> GC <number> Number from GC1 <expDate> 1221 <cardValidationNum> 123		Activate Response: <response> 217 <message> Already active <availableBalance> 14000	

TABLE 2-27 Private Label Gift Card Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
GC16	Authorization: <amount> 1500000 <type> GC <number> Number from GC1 <expDate> 1221 <cardValidationNum> 123		Authorization Response: <response> 352 <message> Decline <cardValidationResult> M <availableBalance> 14000	
GC17	Authorization: <amount> 15000 <type> GC <number> Supplied by Implementation Consultant <expDate> 1221 <cardValidationNum> 123		Authorization Response: <response> 281 <message> Card not active	
GC19	Sale: <amount> 1000 <type> GC <number> Supplied by Implementation Consultant <expDate> 1220 <cardValidationNum> 123		Sale Response: <response> 000 <message> Approved <authCode> 11111 <cardValidationResult> M <availableBalance> 1500	
GC19A	Gift Card Credit: <littleTxnId> Value from GC19 resp. <creditAmount> 10000 <type> GC <number> Value from GC19 <expDate> 1220		Credit Response: <response> 000 <message> Approved Declined Transaction report result = 304 - Lost/Stolen Card	

TABLE 2-27 Private Label Gift Card Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
GC20	Balance Inquiry: <type> GC <number> Number from GC4 <expDate> 0121 <cardValidationNum> 476		Balance Inquiry Response: <response> 352 <message> Invalid CVV2	

2.5.16 Testing Mobile Point of Sale

In addition to other normal required certification testing, you should submit the following transactions to test the Mobile Point of Sale offering. Because these test cases require you to submit specific values, they are not end-to-end test (i.e., mobile device to response message). They are intended to verify your system's handling of the various response codes and messages associated with the mobile solution.

To test end-to-end operation swipe a test card. Is long as the LittleXML message and the submitted data is valid the certification system returns a response message indicating that the transaction was approved.

NOTE: These test cases require the substitution of specific values for the `<encryptedTrack>` element. If a value for a required element is not provided, submit any valid value.

Submission of any transactions not using the specified values will result in an approved response, as long as the message contains valid data and the `track1Status` and `track2Status` elements are set to 0.

1. Using the data provided in [Table 2-28](#) for orderId mpos_auth1, submit a Sale transaction. Verify that the response data matches the values for the Key Response Elements for that orderId.
2. Using the data provided in [Table 2-28](#) for orderId mpos_credit, submit a Credit transaction. Verify that the response data matches the values for the Key Response Elements.
3. Using the data provided in [Table 2-28](#) for orderId mposAuthFailure1, submit a Sale transaction. This transaction simulates the failure to properly read the Track1 and Track 2 data. Verify that the response data matches the values for the Key Response Elements and that your system properly handles this response.
4. Using the data provided in [Table 2-28](#) for orderId mposAuthFailure2, submit a Sale transaction. This transaction simulates a timeout error encountered while waiting for the decrypted card information. Verify that the response data matches the values for the Key Response Elements and that your system properly handles this response.

5. Using the data provided in [Table 2-28](#) for orderId mposAuthFailure3, submit a Sale transaction. This transaction simulates a failure to decrypt the card information. Verify that the response data matches the values for the Key Response Elements and that your system properly handles this response.

NOTE: When in production, if you receive the soft declines in Steps 4 and 5, the best practice would be to re-swipe the card and resubmit the transactions.

TABLE 2-28 Mobile Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
mpos_sale	<encryptedTrack> <ksn> <formatId> <track1Status> <track2Status>	CASE1... (see below) 77853211300008E0 0016 30 0 0	<response> <message> <authCode> <avsResult>	000 Approved 22222 34
mpos_credit	<encryptedTrack> <ksn> <formatId> <track1Status> <track2Status>	CASE1... (see below) 77853211300008E0 0016 30 0 0	<response> <message>	000 Approved
mposAuthFailure1	<encryptedTrack> <ksn> <formatId> <track1Status> <track2Status>	CASE8... (see below) 77853211300008E0 0016 30 1 1	<response> <message>	524 Hard Decline - Input data is invalid

TABLE 2-28 Mobile Test Data (Continued)

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
mposAuthFailure2	<encryptedTrack> <ksn> <formatId> <track1Status> <track2Status>	TimeOut 0016 30 0 0	<response> <message>	521 Soft decline - Card reader decryption service is not available
mposAuthFailure3	<encryptedTrack> <ksn> <formatId> <track1Status> <track2Status>	CASE2...(see below) 0016 30 0 0	<response> <message>	523 Soft decline - Decryption failed

CASE1 encryptedTrack value -

CASE1E185EADD6AFE78C9A214B21313DCD836FDD555FBE3A6C48D141FE80AB9172B9
 63265AFF72111895FE415DEDA162CE8CB7AC4D91EDB611A2AB756AA9CB1A000000000
 00000000000000000000000000000005A7AAF5E8885A9DB88ECD2430C497003F2646619A2382FFF20
 5767492306AC804E8E64E8EA6981DD

CASE2 encryptedTrack value -

CASE2E185EADD6AFE78C9A214B21313DCD836FDD555FBE3A6C48D141FE80AB9172B9
 63265AFF72111895FE415DEDA162CE8CB7AC4D91EDB611A2AB756AA9CB1A000000000
 00000000000000000000000000000005A7AAF5E8885A9DB88ECD2430C497003F2646619A2382FFF20
 5767492306AC804E8E64E8EA6981DD

CASE8 encryptedTrack value -

CASE8E185EADD6AFE78C9A214B21313DCD836FDD555FBE3A6C48D141FE80AB9172B9
 63265AFF72111895FE415DEDA162CE8CB7AC4D91EDB611A2AB756AA9CB1A000000000
 00000000000000000000000000000005A7AAF5E8885A9DB88ECD2430C497003F2646619A2382FFF20
 5767492306AC804E8E64E8EA6981DD

2.5.17 Testing MasterPass Transactions

If you are planning to support MasterPass transaction, in addition to other required certification testing, you should submit the following transactions to test the use of the additional LittleXML elements required for a MasterPass transaction.

To test the MasterPass functionality using the data supplied in [Table 2-29](#):

1. Submit an Authorization transaction using the data supplied for Order Id MCWalletAuth. Verify that the response data matches the values for the Key Response Elements for that orderId.
2. Submit a Sale transaction using the data supplied for Order Id MCWalletSale. Verify that the response data matches the values for the Key Response Elements for that orderId.

TABLE 2-29 MasterPass Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
MCWalletAuth	<wallet> <walletSourceType> <walletSourceTypeId>	(parent element) MasterPass 102	<response> <message>	000 Approved
MCWalletSale	<wallet> <walletSourceType> <walletSourceTypeId>	(parent element) MasterPass 102	<response> <message>	000 Approved

2.5.18 Testing Apple Pay Transaction Processing

The following Apple Pay tests allow you to verify your submission of valid XML for two Apple Pay scenarios. The first test scenario applies to the case where you decrypt the Apple Pay PKPaymentToken, while the second scenario applies to the Vantiv decryption PKPaymentToken components submitted in a LittleXML transaction.

NOTE:

For information about testing the submission of the PKPaymentToken using the Mobile eProtect, please refer to the *Vantiv eProtect Integration Guide*.

2.5.18.1 Testing the Submission of the Decrypted PKPaymentToken in LittleXML

To test the submission of an Apple Pay transaction in LittleXML when you decrypt the PKPaymentToken, you must include the information listed in the table below in an Authorization or Sale transaction. Assuming you submit valid XML with appropriate values, the test environment will return an approved response message.

TABLE 2-30 LittleXML Elements for Merchant Decrypted PKPaymentToken

LittleXML Element	Value
number	Use any valid (Mod-10 compliant) card number. You can find test numbers in Chapter 2 of the <i>Vantiv LittleXML Reference Guide</i> .
expDate	Use any valid expiration date (i.e., a date in the future).
authenticationValue	Any Base-64 encoded value between 40 and 56 characters in length. This value simulates the cryptogram extracted from the PKPaymentToken.
orderSource	Set to applepay

2.5.18.2 Testing the Submission of PKPaymentToken in LittleXML

When you receive a PKPaymentToken, you must parse (not decrypt) the component parts and use the data to populate the LittleXML <applepay> child elements. This test is designed to allow you to verify your ability to parse the PKPaymentToken data and construct well formed LittleXML. The test uses the data from the example PKPaymentToken shown below (data element names in bold red type) to construct an Authorization or Sale transaction (see [applepay](#)). The system returns a Response Code of 000 - Approved.

Example: PKPaymentToken

```
{
  "version": "EC_v1",
  "data": "Ww9EI+10VVpyZrAb3nxu9c8PG4JEhIh4oTkDhZi4axj5QqC5WIir6
  TJcFmk/3wkrNL/KaRXz3aan4WR06cPL+cUTRpQU09EcqTBTImQbjxGbn42713TyI+y97k3ms17bd5rJ0
  MIOpkCtfp2ua+3lnBhjGFnUzdCxq+/K6eoIEwYlAEfx9Sdpjm+p1VfvSK7vj0BQCcXo1dXGkNUKwKWA4
  GYPUE3qwbu1QWcZwLxAEF43274pACV4LBmdv0HYgYpcgCY0+U6/YSVKdpPrhHLDeL0107T4WwuimHojs
  KA/BknpPY1uJfP+YJxj1fYghaAOAR0tA5cYJft1WLXaV91Zu113Ns1rxColh4PR8wsuw81Cd0ruvoURG
  NaNyX+hG1suQoHeE8ECzKIE6D1HEEVmcdxXYsWFPY7hxEY1QKSSXw==",
  "signature": "MIAGCSqGSI
  b3DQEHAqCAMIACAQExDzANBglghkgBZQMEAgEFADCABgkqhkiG9w0BbwEAAKCAMIIICvzCCAmWgAwIBAg
  IIQpCV6UIIb4owCgYIKoZIzj0EAwIwejEuMCwGA1UEAwqlQXBwbGUgQXBwbGjYXRpb24gSW50ZWdyYX
  Rpb24gQ0EgLSBHMzEmMCQGA1UECwwdQXBwbGUgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkxEzARBgNVBA
  oMCkFwcGx1IEluYy4xCzAJBgNVBAYTA1VTMB4XDTE0MDUwODAxMjMzOVoXDTE5MDUwNzAxMjMzOVowXz
  E1MCMGA1UEAwwcZWNjLXNtcc1icm9rZXItc2lnb19VQzQtUFJPRDEUMBIGA1UECwwLaU9TIFN5c3RlbX
  MxEzARBgNVBAoMCkFwcGx1IEluYy4xCzAJBgNVBAYTA1VTMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQg
  AEwhV37evWx7Ihj2jdcJChIY3HsL1vLCg9hGCV2Ur0pUEbg0I02BhzQH6DMx8cVMP36zIg1rrV10/0ko
  mJPnwPE60B7zCB7DBFBgggrBqEFBQcBAQQ5MDcwNQYIKwYBBQUHMGAGKWh0dHA6Ly9vY3NwLmFwcGx1Lm
  NvbS9vY3NwMDQtYXBwbGUhaWNhMzAxMB0GA1UdDgQWBBUV9tv1XSBlomJdi9+v4UH5tYJDAMBgNVHR
  MBAf8EAjaAMB8GA1UdIwQYMBaAFCPyScRPk+TvJ+bE9ihsP6K7/S5LMDQGA1UdHwQtMCswKaAnoCWGI2
  h0dHA6Ly9jcmwuYXBwbGUuY29tL2FwcGx1Yw1jYTmuY3JsMA4GA1UdDwEB/wQEAvIHgDAPBqkqhkiG92
  NkBh0EAQgUAMAoGCCqGSM49BAMCA0gAMEUCIQCqGdtAk+7wXrBV7jTwzCBLE+OcrVL15hjf0relJiPGg
  IgXGHYYeXwrn02Zwc15TT1W8rIqK0QuIvOn01THCbkhVowggLuMIICdaADAgECAgjhB+S+/OpjalzAKBq
  gqhkjOPQQDAjBnMRswGQYDVQQDBJBcHbsZSBs290IENBIC0gRzMxJjAkBgNVBAsMHUFwcGx1IENlcn
  RpZmljYXRpb24gQXV0aG9yaXR5MRMwEQYDVQQDApBcHBsZSBjbmMuMQswCQYDVQQGEwJVUzAeFw0xDN
```

```
A1MDYyMzQ2MzBaFw0yOTA1MDYyMzQ2MzBaMHoxLjAsBgNVBAMMJUFwcGx1IEFwcGxpY2F0aW9uIEludGVncmF0aW9uIENBIC0gRzMxJjAkBgNVBAsMHUFwcGx1IEN1cnRpZmljYXRpb24gQXV0aG9yaXR5MRMwEQYDVQQKDApBcHBsZSBjbmMuMQswCQYDVQQGEwJVUzBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABPAXEYQZ12SF1RpeJYEHduiAou/ee65N4I38S5PhM1bVzlslriLQl3YNIk57ugj9dhf0iMt2u2ZwvsjOKYT/VEWjgfcwgfQwRgYIKwYBBQUHAQEEoja4MDYGCCsGAQUFBzABhipodHRwOi8vb2NzcC5hcHBsZS5jb20vb2NzcDA0LWFwcGx1cm9vdGNhZzMwHQYDVR0OBByEFCPyScRPk+Tvj+bE9ihsP6K7/S5LMA8GA1UdEwEB/wQFMAMBAf8wHwYDVR0jBBgwFoAUu7DeoVgziJqkipnevr3rr9rLJKswNwYDVR0fBDawLjAsocggKIYmaHR0cDovL2Nybc5hcHBsZS5jb20vYXBwbGVyb290Y2FnMy5jcmwwDgYDVR0PAQH/BAQDAgEGMBAGCiqGSIb3Y2QGAg4EAguAMAOGCCqGSM49BAMCA2cAMGQCMDrPcoNRFPmxhvs1wlBKyr/0F+3ZD3VNoo6+8ZyBXkK3ifiY95tZn5jVQQ2PnenC/gIwMi3VRCGwowV3bF3zODuQZ/0XfCwhbZZPxNjpghJvVPh6fRuZy5sJiSFhBpkPCZIdAAAxggFFMIIBWwIBATCBhjb6MS4wLAYDVQQDCBVcHBsZSBBCBhsaWNhdGlvbiBJbnR1Z3JhdGlvbiBDQSAtIEczMSYwJAYDVQQLDB1BcHBsZSBZXJ0aWZpY2F0aW9uIEF1dGhvcmloetETMBEGA1UECgwKQXBwbGUgSW5jLjELMAkGA1UEBhMCVVMCEKQlelCCG+KMA0GCWCGSAFlAwQCAQUAoGkwGAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEhATAcBgkqhkiG9w0BCQUxDxcNMTQxMDA2MjE1NjQzWjAvBkgqhkig9w0BCQQxIgQgg8i4X6yRAU7AXS11amCf02UIQlpUvNPToXuaamsFUT8wCgYIKoZIZj0EAwIERzBFaibe17NGTuuk+W901k30ac4Z90PoMhNIqRqnij9KNEb/XAIhALELZyDWw0fQM8t0px086gg9xFz424rEM1J01TM1VxhAAAAAAA", "header": { "applicationData": "496461ea64b50527d2d792df7c38f301300085dd463e347453ae72debf6f4d14", "transactionId": "f9b0d3cfbb64cd155249c691aca3c521de03725720616b810d90341f97f347b7", "ephemeralPublicKey": "MFkwEwYHKoZIZj0CAQYIKoZIZj0DAQcDQgAEarp8x0hLX9QliUPS9c54i3cqEfrJD37NG75ieNxncOeFLkjCk", "publicKeyHash": "jAMRQqg4nffHrXvfwRfbaEc11bk3QD3rv5K9xLqLgu0=" } }
```

2.5.19 Testing Android Pay Transaction Processing

The certification testing you need to perform depends upon the integration method you implement. This section provides information about the required tests for the eProtect integration method.

2.5.19.1 Testing Android Pay using eProtect

In this integration method, your application submits information to the Google test environment, receives a low-value token, and then submits the low value token to Vantiv in a normal payment transaction.

The table below provides key data values you provide in the submitted transaction.

NOTE:	At this time, we are unable to support end-to-end testing in our Pre-Live environment for Android Pay using eProtect option. Low-value tokens returned from Google are generated independently by their test environment. As a result, all transactions submitted using a low-value token returned by Google result in a response message containing the Reason Code 878 - Expired PayPage Registration Id.
	These test allow you to verify your submission of well formed XML conforming to the schema files.

1. Perform an implicit token registration by submitting a Sale or Authorization transaction using the data shown for the androidpay_1 (orderId) transaction in Table 1.
2. Verify the response message includes the key Response values shown.
3. Perform an explicit token registration by submitting a Register Token Request using the data shown for the androidpay_2 (orderId) transaction in Table 1.
4. Verify the response message includes the key Response values shown.

TABLE 2-31 Android Pay using eProtect Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
androidpay_1	<orderSource> <paypageRegistrationId>	androidpay Value returned from Google	<response> <message>	878 Expired Paypage registration Id
androidpay_2	<paypageRegistrationId>	Value returned from Google	<response> <message>	878 Expired Paypage registration Id

Example: Implicit Token Registration via Authorization Transaction

```

<littleOnlineRequest version="10.2" 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>androidpay_1</orderId>
    <amount>40000</amount>
    <orderSource>androidpay</orderSource>
    <paypage>
      <paypageRegistrationId>Low-Value Token</paypageRegistrationId>
    </paypage>
  </authorization>
</littleOnlineRequest>

```

Example: Explicit Token Registration Transaction

```
<littleOnlineRequest version="10.2" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <registerTokenRequest id="99999" reportGroup="RG1">
    <orderId>androidpay_2</orderId>
    <paypageRegistrationId>Low-Value Token</paypageRegistrationId>
  </registerTokenRequest>
</littleOnlineRequest>
```

2.5.20 Testing SEPA Direct Debit Transaction

Currently, the SEPA Direct Debit certification tests cover the Vantiv supplied Mandate scenario only. End-to-end testing involves simulating a consumer on your test site, who selects SEPA Direct Debit as the payment method and approves the Mandate.

To test the SEPA Direct Debit alternate payment method, do the following:

1. Construct a Sale transaction using the information provided for orderId 1_sddSale, and submit it to the Pre-Live environment. Ideally, you should do this using a test checkout page as a simulated consumer. Make sure you include all required fields, as well as the name, email, and country child elements of billToAddress.
2. Redirect the customer browser to the redirectUrl from the response message. Also, record the redirectToken and the mandateReference values.
3. As the customer, accept the Mandate. After redirecting the browser back to your checkout page, verify that the redirectToken parameter from the URL matches the redirectToken value from the response message.
4. To simulate a recurring (subsequent) payment, construct a Sale transaction using the information provided for orderId 1a_sddSale, including the mandateReference value returned in the 1_ssdSale response message.
5. Verify the approval of the transaction.
6. Construct a Sale transaction using the information provided for orderId 15_sddSale, and submit it to the Pre-Live environment.
7. Verify you receive a 904 response code and that you handle it appropriately. In a production environment you should prompt the consumer to reenter their IBAN.
8. Construct a Sale transaction using the information provided for orderId 24_sddSale, and submit it to the Pre-Live environment.

9. Verify you receive a 379 response code and that you handle it appropriately. In a production environment you should prompt the consumer to select a different payment method.
10. Construct a Sale transaction using the information provided for orderId 2_sddSale, and submit it to the Pre-Live environment.
11. Verify you receive a 386 response code and that you handle it appropriately. In a production environment you should retry submitting the transaction.

TABLE 2-32 SEPA Direct Debit Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
1_sddSale	<name> <mandateProvider> <sequenceType> <iban>	David Berman Vantiv FirstRecurring DE7985050300310 0180568	<response> <message> <redirectUrl> <redirectToken> <mandateReference>	000 Approved Cert Mandate page Dynamically Generated Dynamically Generated
1a_sddSale	<name> <mandateProvider> <sequenceType> <mandateReference> <iban>	David Berman Vantiv SubsequentRecurring Value from 1_sddSale DE7985050300310 0180568	<response> <message>	000 Approved
15_sddSale	<name> <mandateProvider> <sequenceType> <iban>	David Berman Vantiv OneTime DE79850503A0310 0180568	<response> <message>	904 Invalid IBAN
24_sddSale	<name> <mandateProvider> <sequenceType> <iban>	girogateDecline Vantiv OneTime DE79850503A0310 0180568	<response> <message>	379 Transaction declined by the processing network

TABLE 2-32 SEPA Direct Debit Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
2_sddSale	<name> <mandateProvider> <sequenceType> <iban>	unknownError Vantiv OneTime DE7985050300310 0180568	<response> <message>	386 Processing Network Error

2.5.21 Testing iDEAL Transactions

iDEAL (Real-time Bank Transfers) is an international alternate payment method used in the Netherlands.

To test the iDEAL alternate payment method, do the following:

1. Construct a Sale transaction using the information provided for orderId p1_idealSale and submit it to the Pre-Live environment. Ideally, you should do this using a test checkout page as a simulated consumer. Make sure you include all required fields and set the (billing) country to NL.
2. Redirect the customer browser to the redirectUrl from the response message. Also, record the redirectToken.
3. As the customer, accept the agreement. After redirecting the browser back to your checkout page, verify that the redirectToken parameter from the URL matches the redirectToken value from the response message.

NOTE: The agreement page presented by the Pre-Live simulator differs from the bank page that a consumer uses and may vary by bank. Although the mechanics of the consumer accepting the agreement may differ, any difference does not change the operations from the merchants standpoint.

4. Verify the approval of the agreement by comparing the redirectToken value to the token_value parameter returned in the URL.
5. Construct a Sale transaction using the information provided for orderId n10_idealSale and submit it to the Pre-Live environment.
6. Verify that the transaction fails with a Response Code of 917 - Invalid Country and that your system handles the response correctly. All iDeal transaction must use NL as the (Billing) country.

TABLE 2-33 iDEAL Test Data

orderId	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
p1_idealSale	<amount> <name> <country>	10011 David Berman NL	<response> <message> <redirectUrl> <redirectToken>	000 Approved Cert bank page Dynamically Generated
n10_idealSale	<amount> <name> <country>	20100 David Berman USA	<response> <message>	917 Invalid Billing Country

2.5.22 Testing Online Duplicate Transaction Processing

When you submit certain Online transactions, the system acts to detect if it is a duplicate by comparing the `id` attribute and the credit card number against other successful Online transactions of the same type processed within the previous two days. The system performs this checking routine for the following transaction types: Capture, Force Capture, Capture Given Auth, Credit, Sales, eCheck Credit, eCheck Sales, eCheckVoid, and Void, as well as Gift Card transactions.

If the system determines a transaction to be a duplicate, The transaction appears in the Declined Transaction report. This report is available in near real-time via Vantiv iQ, and as an Secure Scheduled Report, which is generated daily for the previous days transactions. Please refer to [Online Duplicate Checking](#) on page 8 for additional information.

To test your handling of duplicate transactions:

NOTE: When you submit the duplicate transaction, make sure that all information, including the `id` attribute, is identical.

- Send any of the following Capture transactions more than once within a two day period: Order numbers 1A, 2A, 3A, 4A, or 5A. The second submission will appear in the Declined Transaction report with a response Reason Code of 251 - Duplicate Transaction. Note: You may have to submit the corresponding Authorization transaction prior to submitting the Capture transaction.
- Send any of the following Credit transactions more than once within a two day period: Order numbers 1B, 2B, 3B, 4B, or 5B. The second submission will appear in the Declined

- Transaction report with a response Reason Code of 251 - Duplicate Transaction. Note: You may have to submit the corresponding Capture transaction prior to submitting the Credit transaction.
3. Send any of the following Void transactions more than once within a two day period: Order numbers 1C, 2C, 3C, 4C, or 5C. The second submission will appear in the Declined Transaction report with a response Reason Code of 251 - Duplicate Transaction. Note: You may have to submit the corresponding Sale transaction prior to submitting the Void transaction.
 4. Send either of the following eCheck Sale transactions more than once within a two day period: Order numbers 42 or 43. The second submission will appear in the Declined Transaction report with a response Reason Code of 251 - Duplicate Transaction.
 5. Send any of the following eCheck Credit transactions more than once within a two day period: Order numbers 46, 47, or 48. The second submission will appear in the Declined Transaction report with a response Reason Code of 251 - Duplicate Transaction.

2.5.23 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 LittleXML transaction messages. Where differences exist between the structure of Batch and Online transactions, the sections present examples of both structures.

NOTE: In the LittleXML, the order of the elements is enforced. Failure to adhere to the element order as defined in the schema will result in XML validation errors.

This chapter discusses the following topics:

- Overview of Online and Batch Processing Formats
- Transaction Types and Examples

NOTE: This chapter does not include examples of the PayFac Instruction-Based Dynamic Payout transaction types. For additional information about these transactions, please refer to [Appendix D, "PayFac™ Dynamic Payout".](#)

3.1 Overview of Online and Batch Processing Formats

There are two methods of submitting payment transactions using the LittleXML 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

We treat each Batch transmission you send to us as 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 retrieval request for the response to a previously processed session. We return a response transmission for each request.

Batch processing supports the following transaction types:

- Activate Transactions
- Authorization Transactions
- Authorization Reversal
- Balance Inquiry Transactions
- Cancel Subscription Transactions
- Capture Transactions
- Capture Given Auth Transactions
- Create Plan Transactions
- Credit Transactions
- Deactivate Transactions
- eCheck Credit Transactions
- eCheck Prenotification Credit
- eCheck Prenotification Sale
- eCheck Redeposit Transactions
- eCheck Sale Transactions
- eCheck Verification Transactions
- Force Capture Transactions
- Fraud Check Transaction
- Gift Card Auth Reversal Transaction
- Gift Card Capture Transaction
- Gift Card Credit Transaction
- Load Transactions
- Register Token Transactions
- RFR Batch Transactions (Batch Only)
- Sale Transactions
- Unload Transactions
- Update Card Validation Number Transactions
- Update Plan Transactions
- Update Subscription Transactions
- All PayFac Dynamic Payout Transactions

NOTE: For information and examples of Dynamic Payout Funding Instructions, please refer to [Appendix D, "PayFac™ Dynamic Payout"](#).

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

3.1.1.1 Supported Communication Protocols

We supports the following communication protocols for Batch processing:

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

For additional information concerning the recommended transmission methods, see [Transferring Session Files](#) on page 95.

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

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).

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.).

NOTE: For information on the XML Validation response and message attributes, please refer to [XML Validation Error Messages](#) on page 859.

3.2 Online Processing Format

Each Online request you send to us is a single transaction. We process Online transactions upon receipt and return a response file.

Online processing supports the following transaction types:

- Activate Transactions
- Activate Reversal Transactions (Online Only)
- Authorization Transactions
- Authorization Reversal
- Balance Inquiry Transactions
- Cancel Subscription Transactions
- Capture Transactions
- Capture Given Auth Transactions
- Create Plan Transactions
- Credit Transactions
- Deactivate Transactions
- Deactivate Reversal Transactions (Online Only)
- Deposit Reversal Transactions (Online Only)
- Force Capture Transactions
- Load Transactions
- Load Reversal Transactions (Online Only)
- eCheck Sale Transactions
- eCheck Credit Transactions
- eCheck Redeposit Transactions
- eCheck Verification Transactions
- eCheck Void Transactions (Online Only)
- Refund Reversal Transactions (Online Only)
- Sale Transactions
- Status Query Transaction (Online Only)
- Unload Transactions
- Unload Reversal Transactions (Online Only)
- Update Card Validation Number Transactions
- Update Plan Transactions

- Update Subscription Transactions
- Void Transactions (Online Only)

3.2.1 Supported Communication Protocols

Title & Co. supports the HTTPS POST communication protocol for Online processing.

For additional information concerning the recommended transmissions methods, see [Transferring Online Files](#) on page 96.

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

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

NOTE: For information on the XML Validation response and message attributes, please refer to [XML Validation Error Messages](#) on page 859.

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:

NOTE: **In the LittleXML, the order of the elements is enforced. Failure to adhere to the element order as defined in the schema will result in XML validation errors.**

- [Authorization Transactions](#)
- [Authorization Reversal Transactions](#)
- [Activate Transactions](#) (Private Label Gift Card transaction)
- [Activate Reversal Transactions \(Online Only\)](#) (Private Label Gift Card transaction)
- [Balance Inquiry Transactions](#) (Private Label Gift Card transaction)
- [Cancel Subscription Transactions](#) (Recurring Engine transaction)
- [Capture Transactions](#)
- [Capture Given Auth Transactions](#)
- [Create Plan Transactions](#) (Recurring Engine transaction)
- [Credit Transactions](#)
- [Deactivate Transactions](#) (Private Label Gift Card transaction)
- [Deactivate Reversal Transactions \(Online Only\)](#) (Private Label Gift Card transaction)
- [Deposit Reversal Transactions \(Online Only\)](#) (Private Label Gift Card transaction)
- [eCheck Credit Transactions](#)
- [eCheck Prenotification Credit Transactions \(Batch Only\)](#)
- [eCheck Prenotification Sale Transactions \(Batch Only\)](#)

- [eCheck Redeposit Transactions](#)
- [eCheck Sale Transactions](#)
- [eCheck Verification Transactions](#)
- [eCheck Void Transactions \(Online Only\)](#)
- [Force Capture Transactions](#)
- [Fraud Check Transaction](#)
- [Gift Card Auth Reversal Transactions \(Private Label Gift Card transaction\)](#)
- [Gift Card Capture Transactions \(Private Label Gift Card transaction\)](#)
- [Gift Card Credit Transactions \(Private Label Gift Card transaction\)](#)
- [Load Transactions \(Private Label Gift Card transaction\)](#)
- [Load Reversal Transactions \(Online Only\) \(Private Label Gift Card transaction\)](#)
- [Status Query Transactions \(Online Only\)](#)
- [Refund Reversal Transactions \(Online Only\) \(Private Label Gift Card transaction\)](#)
- [Register Token Transactions](#)
- [RFR Transactions \(Batch Only\)](#)
- [Sale Transactions](#)
- [Unload Transactions \(Private Label Gift Card transaction\)](#)
- [Unload Reversal Transactions \(Online Only\) \(Private Label Gift Card transaction\)](#)
- [Update Plan Transactions \(Recurring Engine transaction\)](#)
- [Update Subscription Transactions \(Recurring Engine transaction\)](#)
- [Update Card Validation Number 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
Discover	10 days
MasterCard	7 days
PayPal	29 days total; Vantiv recommends three days. For more information about PayPal authorizations, see the <i>Vantiv 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 us.

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="iQ Report Group"
customerID="Customer Id">

    <orderId>Order Id</orderId>
    <amount>Authorization Amount</amount>
    <secondaryAmount>Secondary Amount</secondaryAmount>
    <orderSource>Order Entry Source</orderSource>
    <customerInfo>
        <billToAddress>

```

```

<shipToAddress>
[ <card> | <paypal> | <paypage> | <token> | <mpos> | <applepay>]
<billMeLaterRequest>
<cardholderAuthentication>
<processingInstructions>
<pos>
<customBilling>
<taxType>payment or fee</taxType>
<enhancedData>
<amexAggregatorData>
<allowPartialAuth>
<healthcareIIAS>
<filtering>
<merchantData>
<recyclingRequest> (for Recovery merchants using Recycling)
<fraudFilterOverride>
<recurringRequest> (for Recurring Engine merchants)
<debtRepayment>true or false</debtRepayment>
<advancedFraudChecks>
<wallet>
<processingType>processingType Enum</processingType>
<originalNetworkTransactionId>Network Txn Value</originalNetworkTransactionId>
<originalTransactionAmount>Amount from Orig Txn</originalTransactionAmount>
</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="11.0" xmlns="http://www.little.com/schema"
  numBatchRequests = "1">
  <authentication>
    <user>userName</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@vantiv.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@vantiv.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>
</littleRequest>
```

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.

```
<littleRequest version="11.0" xmlns="http://www.little.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>completetread</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>400555000081019</number>
                <expDate>0911</expDate>
            </card>
            <pos>
                <capability>keyedonly</capability>
            </pos>
        </authorization>
    </batchRequest>
</littleRequest>
```

```

<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, we return 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="11.0" 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>4000000000000002</number>
      <expDate>1209</expDate>
      <cardValidationNum>555</cardValidationNum>
    </card>
  </authorization>
</littleOnlineRequest>

```

```

</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](#).

NOTE: When you submit the CVV2/CVC2/CID in a registerTokenRequest, the platform encrypts and stores the value on a temporary basis for later use in a tokenized Auth/Sale transaction submitted without the value. To use the store value when submitting an Auth/Sale transaction, set the cardValidationNum value to 000.

```

<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@vantiv.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@vantiv.com</email>
  </shipToAddress>

```

```

</shipToAddress>
<token>
  <littleToken>1111000101039449</littleToken>
  <expDate>1112</expDate>
  <cardValidationNum>987</cardValidationNum>
</token>
</authorization>

```

Example: Authorization Request with Recurring Request

```

<littleOnlineRequest version="11.0" 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>4000000000000002</number>
      <expDate>1209</expDate>
      <cardValidationNum>555</cardValidationNum>
    </card>
    <cardholderAuthentication>
      <authenticationValue>BwABBJQ1gJDUCAAAAAA=</authenticationValue>
      <authenticationTransactionId>gMV75TmjAgk=</authenticationTransactionId>
    </cardholderAuthentication>
    <recurringRequest>
      <subscription>
        <planCode>Gold_Monthly_1</planCode>
        <numberOfPayments>12</numberOfPayments>
      </subscription>
    </recurringRequest>
  </authorization>
</littleOnlineRequest>

```

```

<startDate>2016-07-21</startDate>
<amount>1500</amount>
<createDiscount>
  <discountCode>New_Customer_Discount_1</addOnCode>
  <name>New_Customer</name>
  <amount>750</amount>
  <startDate>2016-07-21</startDate>
  <endDate>2016-07-22</endDate>
</createDiscount>
<createAddOn>
  <addOnCode>Extra_Feature_1</addOnCode>
  <name>Extra_1</name>
  <amount>900</amount>
  <startDate>2016-08-21</startDate>
  <endDate>2016-07-21</endDate>
</createAddOn>
</subscription>
</recurringRequest>
</authorization>
</littleOnlineRequest>

```

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="iQ Report
Group" customerId="Customer Id">

  <littleTxnId>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>
<recycling> (included for declined Auths if feature is enabled)
<recurringResponse> (for Recurring Engine merchants)
<giftCardResponse> (included if Gift Card is Method of Payment)
<applepayResponse>
<cardSuffix>Card Last 4</cardSuffix> (included for ApplePay using VI or MC)
<androidpayResponse>
<networkTransactionId>Txn ID returned from network</networkTransactionId>
</authorizationResponse>

```

Example: Batch Authorization Response

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

```

<littleResponse version="11.0" 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>
                <avssResult>00</avssResult>
            </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>
                <avssResult>00</avssResult>
                <authenticationResult>2</authenticationResult>
            </fraudResult>
        </authorizationResponse>
    </batchResponse>
</littleResponse>

```

```

<fundingSource>
  <type>PREPAID</type>
  <availableBalance>5000</availableBalance>
  <reloadable>NO</reloadable>
  <prepaidCardType>GIFT</prepaidCardType>
</fundingSource>
</enhancedAuthResponse>
</authorizationResponse>
</batchResponse>
</littleResponse>

```

Example: Online Authorization Response including Advanced Fraud Results

NOTE: The online response format contains a <postDate> element, which indicates the date the financial transaction will post.

```

<littleOnlineResponse version="11.0" 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>
      <advancedFraudResults>
        <deviceReviewStatus>pass</deviceReviewStatus>
        <deviceReputationScore>70</deviceReputationScore>
        <triggeredRule>FlashImagesCookiesDisabled</triggeredRule>
      </advancedFraudResults>
    </fraudResult>
    <enhancedAuthResponse>
      <fundingSource>
        <type>PREPAID</type>
        <availableBalance>0</availableBalance>
        <reloadable>YES</reloadable>
        <prepaidCardType>TEEN</prepaidCardType>
      </fundingSource>
    </enhancedAuthResponse>
  </authorizationResponse>
</littleOnlineResponse>

```

```

        </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 reported the new account as closed. Although the Authorization was approved, this information allows you to make an informed decision about how to proceed with the sale.

```

<littleOnlineResponse version="11.0" 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>

```

Example: Online Authorization Response with Recurring Response

```

<littleOnlineResponse version="11.0" 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>2016-07-25T15:13:43</responseTime>
    <postDate>2016-07-25</postDate>
    <message>Approved</message>
  </authorizationResponse>
</littleOnlineResponse>

```

```

<authCode>123457</authCode>
<fraudResult>
  <avsResult>00</avsResult>
  <cardValidationResult>N</cardValidationResult>
  <authenticationResult>2</authenticationResult>
</fraudResult>
<recurringResponse>
  <subscriptionId>1234567890</subscriptionId>
  <responseCode>000</responseCode>
  <responseMessage>Approved</responseMessage>
</recurringResponse>
</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 system. For information on how to use the Authorization Reversal transaction, see [Notes on the Use of Authorization Reversal Transactions](#) on page 76. Also, if you use our Recycling Engine, you can use the authReversal transaction to halt the recycling of an authorization transaction.

NOTE:	To reverse an Authorization on a Closed-loop Gift Card, submit a Gift Card Auth Reversal transaction, not an Authorization Reversal transaction. See Gift Card Auth Reversal Transactions on page 290 for additional information.
--------------	---

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="iQ Report Group"
  customerId="Customer Id">
  <littleTxnId>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="11.0" xmlns="http://www.little.com/schema"
numBatchRequests="1">
<authentication>
<user>userName</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="11.0" 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="iQ Report Group" customerId="Customer Id">
    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
    <responseTime>Date and Time in GMT</responseTime>
    <message>Response Message</message>
</authReversalResponse>
```

Example: Batch Authorization Reversal Response

This example shows a Batch Response containing three Authorization Reversal responses.

```
<littleResponse version="11.0" 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>
            <response>000</response>
            <responseTime>2011-10-14T13:15:43</responseTime>
            <message>Approved</message>
        </authReversalResponse>
        <authReversalResponse id="ID" reportGroup="core" customerId="viRev2">
            <littleTxnId>21200000002809</littleTxnId>
            <response>001</response>
            <responseTime>2011-10-14T13:15:43</responseTime>
            <message>Transaction received</message>
        </authReversalResponse>
        <authReversalResponse id="ID" reportGroup="core" customerId="mcRev3">
            <littleTxnId>21200000002908</littleTxnId>
            <response>001</response>
            <responseTime>2011-10-14T13:15:43</responseTime>
            <message>Transaction received</message>
        </authReversalResponse>
    </batchResponse>
</littleResponse>
```

Example: Online Authorization Reversal Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
```

```

    response="0" message="Valid Format">
<authReversalResponse id="12345" customerId="Customer Id"
reportGroup="Auth Reversals">
    <littleTxnId>12345678</littleTxnId>
    <response>001</response>
    <responseTime>2011-08-30T13:15:43</responseTime>
    <message>Transaction received</message>
</authReversalResponse>
</littleOnlineResponse>

```

3.3.3 Activate Transactions

The Activate transaction changes the status of a (Closed Loop) Gift Card from inactive to active with a value as specified by the amount element. You can also use this transaction type to create a Virtual Gift Card.

NOTE: **You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.**

3.3.3.1 Activate Request

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

```

<activate id="Activate Id" reportGroup="iQ Report Group" customerId="Customer
Id">
    <orderId>Order Id</orderId>
    <amount>Activation Amount</amount>
    <orderSource>Order Entry Source</orderSource>
    [ <card> | <virtualGiftCard> ]
</activate>

```

Example: Activate Request - Gift Card

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
merchantId="100">
    <authentication>
        <user>User Name</user>
        <password>Password</password>
    </authentication>
    <activate id="834262" reportGroup="ABC Division" customerId="038945">

```

```

<orderId>65347567</orderId>
<amount>40000</amount>
<orderSource>ecommerce</orderSource>
<card>
  <type>GC</type>
  <number>9000000000000001</number>
</card>
</activate>
</littleOnlineRequest>

```

Example: Activate Request - Virtual Gift Card

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema">
  <merchantId="100">
    <authentication>
      <user>User Name</user>
      <password>Password</password>
    </authentication>
    <activate id="834262" reportGroup="ABC Division" customerId="038945">
      <orderId>65347567</orderId>
      <amount>40000</amount>
      <orderSource>ecommerce</orderSource>
      <virtualGiftCard>
        <accountNumberLength>16</accountNumberLength>
        <giftCardBin>900000</giftCardBin>
      </virtualGiftCard>
    </activate>
  </littleOnlineRequest>

```

3.3.3.2 Activate Response

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

```

<activateResponse id="Activate Id" reportGroup="iQ Report Group"
  customerId="Customer Id">
  <littleTxnId>Transaction Id</littleTxnId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date transaction posted</postDate> (Online Only)
  <message>Response Message</message>

```

```

<fraudResult>
<giftCardResponse>
<virtualGiftCardResponse>
</activateResponse>

```

Example: Activate Response - Gift Card

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <activateResponse id="834262" reportGroup="ABC Division">
    <littleTxnId>9695064321</littleTxnId>
    <response>001</response>
    <responseTime>2016-11-25T15:13:43</responseTime>
    <postDate>2016-11-25</postDate>
    <message>Transaction received</message>
    <giftCardResponse>
      <txnTime>2016-11-25T15:13:38</txnTime>
      <refCode>123456</refCode>
      <systemTraceId>123456</systemTraceId>
      <sequenceNumber>123456</sequenceNumber>
      <availableBalance>5000</availableBalance>
    </giftCardResponse>
  </activateResponse>
</littleOnlineResponse>

```

Example: Activate Response - Virtual Gift Card

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <activateResponse id="834262" reportGroup="ABC Division">
    <littleTxnId>9695064321</littleTxnId>
    <response>001</response>
    <responseTime>2016-11-25T15:13:43</responseTime>
    <postDate>2016-11-25</postDate>
    <message>Transaction received</message>
    <giftCardResponse>
      <txnTime>2016-11-25T15:13:38</txnTime>
      <refCode>123456</refCode>
      <systemTraceId>123456</systemTraceId>
      <sequenceNumber>123456</sequenceNumber>
      <availableBalance>5000</availableBalance>
    </giftCardResponse>
  </activateResponse>
</littleOnlineResponse>

```

```

<accountNumber>9000000000000001</accountNumber>
<cardValidationNum>978</cardValidationNum>
<pin>1234</pin>
</virtualGiftCardResponse>
</activateResponse>
</littleOnlineResponse>

```

3.3.4 Activate Reversal Transactions (Online Only)

The Activate Reversal transaction changes the status of a newly activated Gift Card from active to inactive, thus reversing the operation of an Active transaction. The Activate Reversal references the associated Activate transaction by means of the `littleTxnId` element returned in the Activate response. This transaction type is available only for Online transactions.

NOTE: You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.

3.3.4.1 Activate Reversal Request

You must structure an Activate Reversal request as shown in the following examples.

```

<activateReversal id="Activate Reversal Id" reportGroup="iQ Report Group"
customerID="Customer Id">

    <littleTxnId>Transaction Id from Activate Response</littleTxnId>
    <card>
        <virtualGiftCardBin>
            <originalRefCode>Reference Code from Activate Response</originalRefCode>
            <originalAmount>Amount from Activate Transaction</originalAmount>
            <originalTxnTime>Transaction Time from Activate Response</originalTxnTime>
            <originalSystemTraceId>Trace Id from Activate Response</originalSystemTraceId>
            <originalSequenceNumber>Seq Num from Activate Rsp</originalSequenceNumber>
        </virtualGiftCardBin>
    </card>
</activateReversal>

```

Example: Activate Reversal Request

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
merchantId="100">
    <authentication>
        <user>User Name</user>

```

```

<password>Password</password>
</authentication>
<activateReversal id="12345" customerId="Customer Id"
reportGroup="Activate Reversals">
<littleTxnId>1234567890123456789</littleTxnId>
<card>
<type>GC</type>
<number>1234102000003558</number>
<cardValidationNum>888</cardValidationNum>
<pin>1234</pin>
</card>
<originalRefCode>123456</originalRefCode>
<originalAmount>1900</originalAmount>
<originalTxnTime>2017-03-21T10:02:46</originalTxnTime>
<originalSystemTraceId>678901</originalSystemTraceId>
<originalSequenceNumber>123456</originalSequenceNumber>
</activateReversal>
</littleOnlineRequest>

```

3.3.4.2 Activate Reversal Response

An Activate Reversal response has the following structure.

```

<activateReversalResponse id="Activate Reversal Id" reportGroup="iQ
Report Group" customerId="Customer Id">
<littleTxnId>Transaction Id</littleTxnId>
<response>Response Code</response>
<responseTime>Date and Time in GMT</responseTime>
<postDate>Date transaction posted</postDate> (Online Only)
<message>Response Message</message>
<giftCardResponse>
</activateReversalResponse>

```

Example: Activate Reversal Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
response="0" message="Valid Format">
<activateReversalResponse id="834262" reportGroup="ABC Division">
<littleTxnId>9695064321</littleTxnId>
<response>000</response>
<responseTime>2017-03-22T15:13:43</responseTime>

```

```

<postDate>2017-03-22</postDate>
<message>Approved</message>
<giftCardResponse>
  <txnTime>2017-03-22T12:00:00</txnTime>
  <refCode>003558</refCode>
  <systemTraceId>834528</systemTraceId>
  <sequenceNumber>123456</sequenceNumber>
  <availableBalance>0</availableBalance>
</giftCardResponse>
</activateReversalResponse>
</littleOnlineResponse>

```

3.3.5 Balance Inquiry Transactions

A Balance Inquiry transaction returns the available balance of a Gift Card.

NOTE: You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.

3.3.5.1 Balance Inquiry Request

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

```

<balanceInquiry id="Balance Inquiry Id" reportGroup="iQ Report Group"
customerId="Customer Id">

  <orderId>Order Id</orderId>
  <orderSource>Order Entry Source</orderSource>
  <card>
</balanceInquiry>

```

Example: Balance Inquiry Request

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <balanceInquiry id="834262" reportGroup="ABC Division"
customerId="038945">

```

```

<orderId>65347567</orderId>
<orderSource>ecommerce</orderSource>
<card>
  <type>GC</type>
  <number>9000000000000001</number>
  <pin>1234</pin>
</card>
</balanceInquiry>
</littleOnlineRequest>

```

3.3.5.2 Balance Inquiry Response

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

```

<balanceInquiryResponse id="Balance Inquiry Id" reportGroup="iQ
Report Group" customerId="Customer Id">
  <littleTxnId>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>
  <fraudResult>
  <giftCardResponse>
</balanceInquiryResponse>

```

Example: Balance Inquiry Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <balanceInquiryResponse id="834262" reportGroup="ABC Division">
    <littleTxnId>9695064321</littleTxnId>
    <orderId>65347567</orderId>
    <response>000</response>
    <responseTime>2017-03-21T15:13:43</responseTime>
    <postDate>2017-03-21</postDate>
    <message>Approved</message>
    <fraudResult>
      <cardValidationResult>N</cardValidationResult>
    </fraudResult>
  </balanceInquiryResponse>
</littleOnlineResponse>

```

```

<giftCardResponse>
    <availableBalance>2000</activateReversalResponse>
</littleOnlineResponse>

```

3.3.6 Cancel Subscription Transactions

The Cancel Subscription transaction cancels the specified subscription, removing the transaction stream managed by the Recurring Engine.

3.3.6.1 Cancel Subscription Request

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

```

<cancelSubscription>
    <subscriptionId>ID of Subscription to Cancel</subscriptionId>
</cancelSubscription>

```

Example: Cancel Subscription Request

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema">
    <merchantId>100</merchantId>
    <authentication>
        <user>User Name</user>
        <password>password</password>
    </authentication>
    <cancelSubscription>
        <subscriptionId>1234</subscriptionId>
    </cancelSubscription>
</littleOnlineRequest>

```

3.3.6.2 Cancel Subscription Response

A Cancel Subscription response has the following structure. The response message is identical for Online and Batch transactions except Online includes the <postDate> element.

```

<cancelSubscriptionResponse>
    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
    <message>Response Message</message>
    <responseTime>Date and Time in GMT</responseTime>

```

```
<subscriptionId>ID of Subscription Canceled</subscriptionId>
</cancelSubscriptionResponse>
```

Example: Cancel Subscription Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <cancelSubscriptionResponse>
        <littleTxnId>1100030055</littleTxnId>
        <response>001</response>
        <message>Transaction received</message>
        <responseTime>2016-07-11T14:48:46</responseTime>
        <subscriptionId>123457</subscriptionId>
    </cancelSubscriptionResponse>
</littleOnlineResponse>
```

3.3.7 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.

NOTE: To perform a capture on a Closed-loop Gift Card, submit a Gift Card Capture transaction, not an Capture transaction. See [Gift Card Capture Transactions](#) on page 292 for additional information.

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.

NOTE: If you settle in a currency other than US or Canada, you cannot use partial captures.

3.3.7.1 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="iQ Report Group" customerId="Customer Id"
```

```

partial="false">

<littleTxnId>Transaction Id</littleTxnId>
<amount>Authorization Amount</amount>
<surchargeAmount>Surcharge Amount</surchargeAmount>
<enhancedData>
<processingInstructions>
<payPalOrderComplete>Set to true for final Capture</payPalOrderComplete>
<pin>Gift Card Pin Number</pin
</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="11.0" 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="11.0" 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>
                    <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>
                </lineItemData>
            </enhancedData>
        </capture>
    </batchRequest>
</littleRequest>
```

```

<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="11.0" xmlns="http://www.little.com/schema">
    <merchantId>100</merchantId>
    <authentication>
        <user>User Name</user>
        <password>password</password>
    </authentication>
    <capture id="2" reportGroup="ABC Division" customerId="038945" partial="false">
        <littleTxnId>13254123434</littleTxnId>
        <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>2011-08-14</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>

```

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="11.0" 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>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>2011-08-14</orderDate>
      <detailTax>
        <taxIncludedInTotal>true</taxIncludedInTotal>
        <taxAmount>55</taxAmount>
        <taxRate>0.0059</taxRate>
        <taxTypeIdentifier>00</taxTypeIdentifier>
        <cardAcceptorTaxId>011234567</cardAcceptorTaxId>
      </detailTax>
    </enhancedData>
  </capture>
</littleOnlineRequest>

```

```

</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>

```

3.3.7.2 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.

```

<captureResponse id="Authorization Id" reportGroup="iQ Report Group"
customerID="Customer Id">

    <littleTxnId>Transaction Id</littleTxnId>
    <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="11.0" 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>
            <response>001</response>
            <responseTime>2016-09-01T10:24:31</responseTime>
            <message>Transaction received</message>
        </captureResponse>
        <captureResponse id="AX54325432" reportGroup="RG12">
            <littleTxnId>84568457</littleTxnId>
            <response>000</response>
            <responseTime>2016-09-01T10:24:31</responseTime>
            <message>message</message>
        </captureResponse>
    </batchResponse>
</littleResponse>

```

Example: Online Capture Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <captureResponse id="2" reportGroup="ABC Division" customerID="038945">
        <littleTxnId>1100030204</littleTxnId>
        <response>001</response>
        <responseTime>2016-07-11T14:48:48</responseTime>
        <postDate>2016-07-11</postDate>
        <message>Transaction received</message>
    </captureResponse>
</littleOnlineResponse>

```

3.3.8 Capture Given Auth Transactions

You typically use a Capture Given Auth transaction when the associated Authorization occurred outside of the 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 system, but the <littleTxnId> is unknown by the submitting party (for example, if the Auth was submitted by a merchant, but a fulfills submits a Capture Given Auth).

3.3.8.1 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="iQ Report Group"
customerID="Customer Id">

<orderId>Order Id</orderId>
<authInformation>
<amount>Authorization Amount</amount>
<secondaryAmount>Secondary Amount</secondaryAmount>
<orderSource>Order Entry Source</orderSource>
<billToAddress>
<shipToAddress>
[ <card> | <token> | <paypage> | <mpos> ]
<customBilling>
<taxType>payment or fee</taxType>
<billMeLaterRequest>
<enhancedData>
<processingInstructions>
<pos>
<amexAggregatorData>
<merchantData>
<debtRepayment>true or false</debtRepayment>
<processingType>processingType Enum</processingType>
<originalNetworkTransactionId>Network Txn Value</originalNetworkTransactionId>
<originalTransactionAmount>Amount from Orig Txn</originalTransactionAmount>
</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="11.0" xmlns="http://www.little.com/schema"
    numBatchRequests="1">
    <authentication>
        <user>User Name</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>
                <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>2011-09-21</orderDate>
                <detailTax>
                    <taxIncludedInTotal>true</taxIncludedInTotal>
                    <taxAmount>55</taxAmount>
                    <taxRate>0.0055</taxRate>
                    <taxTypeIdentifier>00</taxTypeIdentifier>
                </detailTax>
            </enhancedData>
        </captureGivenAuth>
    </batchRequest>
</littleRequest>
```

```

<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>
</titleRequest>

```

Example: Online Capture Given Auth Request

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema">
  <merchantId>100</merchantId>
  <authentication>
    <user>userName</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>PO12345</customerReference>
    </enhancedData>
  </captureGivenAuth>
</littleOnlineRequest>

```

```

<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>

```

3.3.8.2 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.

```

<captureGivenAuthResponse id="Capture Id" reportGroup="iQ Report Group"
  customerId="Customer Id">

  <littleTxnId>Transaction Id</littleTxnId>
  <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="11.0" 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>

```

```

<response>001</response>
<responseTime>2011-09-01T10:24:31</responseTime>
<message>Transaction received</message>
</captureGivenAuthResponse>
</batchResponse>
</littleResponse>

```

Example: Online Capture Given Auth Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema">
  <response>0</response>
  <message>Valid Format</message>
  <captureGivenAuthResponse id="2" reportGroup="ABC Division"
    customerId="038945">
    <littleTxnId>1100030204</littleTxnId>
    <response>001</response>
    <responseTime>2016-07-11T14:48:48</responseTime>
    <postDate>2016-07-11</postDate>
    <message>Transaction received</message>
  </captureGivenAuthResponse>
</littleOnlineResponse>

```

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

```

<captureGivenAuthResponse id="99999" reportGroup="RG1" customerId="444">
  <littleTxnId>21200000022005</littleTxnId>
  <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>

```

3.3.9 Create Plan Transactions

The Create Plan transaction allows you to define several attributes of a recurring payment schedule. Later, you can associate existing, active plans with subscriptions, which establishes a recurring payment schedule managed by the Recurring Engine.

3.3.9.1 Create Plan Request

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

```
<createPlan>
  <planCode>Plan Reference Code</planCode>
  <name>Name of Plan</name>
  <description>Description of Plan</description>
  <intervalType>The Type of Interval</intervalType>
  <amount>Amount of Recurring Payment</amount>
  <numberOfPayments>1 to 99</numberOfPayments>
  <trialNumberOfIntervals>Number of Trial Intervals</trialNumberOfIntervals>
  <trialIntervalType>Type of Trial Period Interval</trialIntervalType>
  <active>true or false</active>
</createPlan>
```

Example: Online Create Plan Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema">
  <merchantId>100</merchantId>
  <authentication>
    <user>User Name</user>
    <password>password</password>
  </authentication>
  <createPlan>
    <planCode>Gold12</planCode>
    <name>Gold_Monthly</name>
    <description>Gold Level with Monthly Payments</description>
    <intervalType>MONTHLY</intervalType>
    <amount>5000</amount>
    <numberOfPayments>4</numberOfPayments>
    <trialNumberOfIntervals>1</trialNumberOfIntervals>
    <trialIntervalType>MONTH</trialIntervalType>
    <active>true</active>
  </createPlan>
</littleOnlineRequest>
```

```
</littleOnlineRequest>
```

3.3.9.2 Create Plan Response

The Create Plan response message is identical for Online and Batch transactions.

```
<createPlanResponse>
  <littleTxnId>Transaction Id</littleTxnId>
  <response>Response Code</response>
  <message>Response Message</message>
  <responseTime>Date and Time in GMT</responseTime>
  <planCode>Plan Reference Code</planCode>
</createPlanResponse>
```

Example: Online Create Plan Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <createPlanResponse>
    <littleTxnId>1100030055</littleTxnId>
    <response>000</response>
    <message>Approved</message>
    <responseTime>2011-07-11T14:48:46</responseTime>
    <planCode>Gold12</planCode>
  </createPlanResponse>
</littleOnlineResponse>
```

3.3.10 Credit Transactions

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

NOTE:	To perform a credit on a Closed-loop Gift Card, submit a Gift Card Credit transaction, not an Credit transaction. See Gift Card Credit Transactions on page 294 for additional information.
--------------	---

- 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.

3.3.10.1 Credit Request for a Vantiv Processed Transaction

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

```
<credit id="Credit Id" reportGroup="iQ Report Group" customerId="Customer Id">
  <littleTxnId>Transaction Id</littleTxnId>
  <amount>Authorization Amount</amount>
  <secondaryAmount>Secondary Amount</secondaryAmount>
  <customBilling>
  <enhancedData>
  <processingInstructions>
    <actionReason>SUSPECT_FRAUD</actionReason>
  </processingInstructions>
</credit>
```

Example: Batch Credit Request for a Vantiv Processed Transaction

To request a Credit against a sale settled by us, you only need to specify the `<littleTxnId>` element. The application uses the `<littleTxnId>` 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.

```
<littleRequest version="11.0" 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="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>
      <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>

```

```

        </detailTax>
    </lineItemData>
</enhancedData>
</credit>
</batchRequest>
</littleRequest>

```

Example: Online Credit Request for a Vantiv Processed Transaction

To request a Credit against a sale settled by us, 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="11.0" 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>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>2011-07-14</orderDate>
            <detailTax>
                <taxIncludedInTotal>true</taxIncludedInTotal>
                <taxAmount>55</taxAmount>
                <taxRate>0.0059</taxRate>
                <taxTypeIdentifier>00</taxTypeIdentifier>
            </detailTax>
        </enhancedData>
    </credit>
</littleOnlineRequest>

```

```

<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>

```

3.3.10.2 Credit Request for a Non-Vantiv Processed Transaction

You must specify the Credit request for a Non-Vantiv 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-Vantiv processed transaction, refunds of this type are not supported for PayPal. If you need to refund non-Vantiv processed transactions and have not maintained a temporary relationship with your former processor for this purpose, please ask your Relationship Manager for alternative options.

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

Example: Batch Credit Request for a Non-Vantiv Processed Transaction

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

```
<littleRequest version="11.0" xmlns="http://www.little.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>
</littleRequest>

```

Example: Online Credit Request for a Non-Vantiv Processed Transaction

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

```

<littleOnlineRequest version="11.0" 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>
    <card>
      <type>VI</type>
    </card>
  </credit>
</littleOnlineRequest>

```

```
<number>4005550000081019</number>
<expDate>0907</expDate>
</card>
<customBilling>
<phone>5555555555</phone>
<descriptor>descriptor</descriptor>
</customBilling>
<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>2011-08-14</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>

```

3.3.10.3 Credit Response

The Credit response message is identical for Online and Batch transactions except Online includes the postDate element.

```

<creditResponse id="Credit Id" reportGroup="iQ Report Group"
customerID="Customer Id">

<littleTxnId>Transaction Id</littleTxnId>
<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="11.0" 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>
<response>001</response>
<responseTime>2016-09-01T10:24:31</responseTime>
<message>Transaction received</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 (`<littleOnlineResponse>` element not shown), so it includes the `<postDate>` element.

```
<creditResponse id="99999" reportGroup="RG1" customerId="444">
  <littleTxnId>21200000473208</littleTxnId>
  <response>001</response>
  <responseTime>2016-10-19T19:29:45</responseTime>
  <postDate>2016-10-19</postDate>
  <message>Transaction received</message>
  <tokenResponse>
    <littleToken>111102200202001</littleToken>
    <tokenResponseCode>801</tokenResponseCode>
    <tokenMessage>Account number was successfully registered</tokenMessage>
    <type>VI</type>
    <bin>402410</bin>
  </tokenResponse>
</creditResponse>
```

3.3.11 Deactivate Transactions

The Deactivate transaction changes the status of a (Closed Loop) Gift Card from active to inactive.

NOTE:	You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.
--------------	---

3.3.11.1 Deactivate Request

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

```
<deactivate id="Activate Id" reportGroup="iQ Report Group" customerId="Customer Id">
  <orderId>Order Id</orderId>
  <orderSource>Order Entry Source</orderSource>
  <card>
```

```
</deactivate>
```

Example: Online Deactivate Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <deactivate id="834262" reportGroup="ABC Division" customerId="038945">
    <orderId>65347567</orderId>
    <orderSource>ecommerce</orderSource>
    <card>
      <type>GC</type>
      <number>9000000000000001</number>
      <pin>1234</pin>
    </card>
  </deactivate>
</littleOnlineRequest>
```

3.3.11.2 Deactivate Response

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

```
<deactivateResponse id="Deactivate Id" reportGroup="iQ Report Group"
  customerId="Customer Id">
  <littleTxnId>Transaction Id</littleTxnId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date transaction posted</postDate> (Online Only)
  <message>Response Message</message>
  <fraudResult>
  <approvedAmount>5000</approvedAmount>
  <giftCardResponse>
</deactivateResponse>
```

Example: Online Deactivate Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
```

```

<deactivateResponse id="834262" reportGroup="ABC Division">
  <littleTxnId>9695064321</littleTxnId>
  <response>000</response>
  <responseTime>2016-07-25T15:13:43</responseTime>
  <postDate>2016-07-25</postDate>
  <message>Approved</message>
  <approvedAmount>5000</approvedAmount>
  <giftCardResponse>
    <availableBalance>0</availableBalance>
    <beginningBalance>5000</beginningBalance>
    <endingBalance>0</endingBalance>
  </giftCardResponse>
</deactivateResponse>
</littleOnlineResponse>

```

3.3.12 Deactivate Reversal Transactions (Online Only)

The Deactivate Reversal transaction changes the status of a newly deactivated Gift Card from inactive to active, thus reversing the operation of an Deactivate transaction. The Deactivate Reversal references the associated Deactivate transaction by means of the littleTxnId element returned in the Deactivate response. This transaction type is available only for Online transactions.

NOTE:	You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.
--------------	--

3.3.12.1 Deactivate Reversal Request

You must structure an Deactivate Reversal request as shown in the following examples.

```

<deactivateReversal id="Deactivate Reversal Id" reportGroup="iQ Report Group"
customerID="Customer Id">

  <littleTxnId>Transaction Id from Load Response</littleTxnId>
  <card>
    <originalRefCode>Reference Code from Deactivate Response</originalRefCode>
    <originalAmount>Amount from Deactivate Transaction</originalAmount>
    <originalTxnTime>Transaction Time from Deactivate Response</originalTxnTime>
    <originalSystemTraceId>Trace Id from Deactivate Resp</originalSystemTraceId>
    <originalSequenceNumber>Seq Num from Deactivate Resp</originalSequenceNumber>

```

```
</deactivateReversal>
```

Example: Deactivate Reversal Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <deactivateReversal id="12345" customerId="Customer Id"
    reportGroup="Deactivate Reversals">
    <littleTxnId>1234567890123456789</littleTxnId>
    <card>
      <type>GC</type>
      <number>1234102000003558</number>
      <cardValidationNum>888</cardValidationNum>
      <pin>1234</pin>
    </card>
    <originalRefCode>123456</originalRefCode>
    <originalAmount>1900</originalAmount>
    <originalTxnTime>2017-03-21T10:02:46</originalTxnTime>
    <originalSystemTraceId>678901</originalSystemTraceId>
    <originalSequenceNumber>123456</originalSequenceNumber>
  </deactivateReversal>
</littleOnlineRequest>
```

3.3.12.2 Deactivate Reversal Response

An Deactivate Reversal response has the following structure.

```
<deactivateReversalResponse id="Deactivate Reversal Id" reportGroup="iQ
Report Group" customerId="Customer Id">
  <littleTxnId>Transaction Id</littleTxnId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date transaction posted</postDate>
  <message>Response Message</message>
  <giftCardResponse>
</deactivateReversalResponse>
```

Example: Online Deactivate Reversal Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <deactivateReversalResponse id="834262" reportGroup="ABC Division">
    <littleTxnId>9695064321</littleTxnId>
    <response>000</response>
    <responseTime>2017-03-22T15:13:43</responseTime>
    <postDate>2017-03-22</postDate>
    <message>Approved</message>
    <giftCardResponse>
      <txnTime>2017-03-22T12:00:00</txnTime>
      <refCode>003558</refCode>
      <systemTraceId>834528</systemTraceId>
      <sequenceNumber>123456</sequenceNumber>
      <availableBalance>5000</availableBalance>
    </giftCardResponse>
  </deactivateReversalResponse>
</littleOnlineResponse>
```

3.3.13 Deposit Reversal Transactions (Online Only)

The Deposit Reversal transaction is a Gift Card transaction that reverses the deposit initiated by either a Capture or Sale transaction. The Deposit Reversal references the associated Capture/Sale transaction by means of the littleTxnId element returned in the Capture/Sale response. This transaction type is available only for Online transactions.

NOTE:	You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.
--------------	---

3.3.13.1 Deposit Reversal Request

You must structure an Deposit Reversal request as shown in the following examples.

```
<depositReversal id="Deposit Reversal Id" reportGroup="iQ Report Group"
  customerId="Customer Id">

  <littleTxnId>Transaction Id from Load Response</littleTxnId>
  <card>
    <originalRefCode>Reference Code from Gift Card Capture Resp</originalRefCode>
    <originalAmount>Amount from Gift Card Capture Resp</originalAmount>
```

```

<originalTxnTime>Transaction Time from GC Capture Response</originalTxnTime>
<originalSystemTraceId>Trace Id from GC Capture Resp</originalSystemTraceId>
<originalSequenceNumber>Seq Num from GC Capture Resp</originalSequenceNumber>
</depositReversal>

```

Example: Deposit Reversal Request

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <depositReversal id="12345" customerId="Customer Id" reportGroup="Deposit
  Reversals">
    <littleTxnId>1234567890123456789</littleTxnId>
    <card>
      <type>GC</type>
      <number>1234102000003558</number>
      <cardValidationNum>888</cardValidationNum>
    </card>
    <originalRefCode>123456</originalRefCode>
    <originalAmount>1900</originalAmount>
    <originalTxnTime>2017-03-21T10:02:46</originalTxnTime>
    <originalSystemTraceId>678901</originalSystemTraceId>
    <originalSequenceNumber>123456</originalSequenceNumber>
  </depositReversal>
</littleOnlineRequest>

```

3.3.13.2 Deposit Reversal Response

An Deposit Reversal response has the following structure.

```

<depositReversalResponse id="Deactivate Reversal Id" reportGroup="iQ
Report Group" customerId="Customer Id">
  <littleTxnId>Transaction Id</littleTxnId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date transaction posted</postDate>
  <message>Response Message</message>
  <giftCardResponse>

```

```
</depositReversalResponse>
```

Example: Online Deposit Reversal Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <depositReversalResponse id="834262" reportGroup="ABC Division">
        <littleTxnId>9695064321</littleTxnId>
        <response>000</response>
        <responseTime>2017-03-22T15:13:43</responseTime>
        <postDate>2017-03-22</postDate>
        <message>Approved</message>
        <giftCardResponse>
            <txnTime>2017-03-22T12:00:00</txnTime>
            <refCode>003558</refCode>
            <systemTraceId>834528</systemTraceId>
            <sequenceNumber>123456</sequenceNumber>
            <availableBalance>5000</availableBalance>
        </giftCardResponse>
    </depositReversalResponse>
</littleOnlineResponse>
```

3.3.14 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 or whether the original transaction was settled by our system, although the requests are structured differently. This section contains the following:

- eCheck Credit Request Against a Vantiv Transaction
- eCheck Credit Request for a Non-Vantiv Processed Sale
- eCheck Credit Response

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

3.3.14.1 eCheck Credit Request Against a Vantiv Transaction

To request an eCheck Credit against an eCheck Sale that had been settled by us, you only need to specify the <littleTxnId> element. When you specify this element, the application uses the <littleTxnId> 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 our system, specify the Credit request as follows:

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

<littleTxnId>Transaction Id</littleTxnId>
<amount>Credit Amount</amount>
<secondaryAmount>Secondary Amount</secondaryAmount>
<customBilling>
<customIdentifier>
</echeckCredit>
```

Example: eCheck Credit Request

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

```
<littleRequest version="11.0" xmlns="http://www.little.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">
<littleTxnId>4455667788</littleTxnId>
<amount>1000</amount>
</echeckCredit>
<echeckCredit reportGroup="new53">
<littleTxnId>4455667789</littleTxnId>
<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>
</titleRequest>

```

3.3.14.2 eCheck Credit Request for a Non-Vantiv Processed Sale

If the original eCheck Sale transaction was not processed via our system, or the if the <littleTxnId> for the original eCheck Sale transaction is not available, specify eCheck Credit request as follows:

```

<echeckCredit id="eCheckCredit Id" reportGroup="iQ Report Group"
customerID="Customer Id">

  <orderId>Order Id</orderId>
  <amount>Credit Amount</amount>
  <secondaryAmount>Secondary Amount</secondaryAmount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
    <echeck> or <echeckToken>
    <customBilling>
    <merchantData>
  </echeckCredit>

```

The third transaction shown in the [eCheck Credit Request](#) example on page 264 shows an example of a Credit request against a non-Vantiv processed transaction.

3.3.14.3 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="iQ Report Group"
customerId="Customer Id">

    <littleTxnId>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>
    <tokenResponse> (for Tokenized merchants submitting account data)
</echeckCreditResponse>
```

Example: eCheck Credit Response

```
<littleResponse version="11.0" 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>
            <response>001</response>
            <responseTime>2016-09-01T10:24:31</responseTime>
            <message>Transaction received</message>
        </echeckCreditResponse>
    </batchResponse>
</littleResponse>
```

3.3.15 eCheck Prenotification Credit Transactions (Batch Only)

You use this non-monetary transaction to verify the consumer's account information prior to submitting an eCheck Credit transaction (also see [eCheck Prenotification](#) on page 47). This transaction type is only supported for US transactions.

3.3.15.1 eCheck Prenotification Credit Request

You must specify the eCheck Prenotification Credit request using the following format:

```
<echeckPreNoteCredit id="eCheck PreNote Id" reportGroup="iQ Report Group"
customerId="Customer Id">

    <orderId>Order Id</orderId>
    <orderSource>Order Entry Source</orderSource>
    <billToAddress>
        <echeck>
            <merchantData>
        </echeck>
    </billToAddress>
</echeckPreNoteCredit>
```

Example: eCheck Prenotification Credit Request

```
<littleRequest version="11.0" xmlns="http://www.little.com/schema" id="123">
    numBatchRequests="1">
        <authentication>
            <user>userName</user>
            <password>password</password>
        </authentication>
        <batchRequest id="654321" numEcheckPreNoteCredit="1" merchantId="100">
            <echeckPreNoteCredit id="AX54321678" reportGroup="RG27" customerId="53">
                <orderId>12z58743y1</orderId>
                <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>
            </echeckPreNoteCredit>
        </batchRequest>
    </littleRequest>
```

```

    </echeck>
    </echeckPreNoteCredit>
  </batchRequest>
</littleRequest>

```

3.3.15.2 eCheck Prenotification Credit Response

The eCheck Prenotification Credit response has the following structure:

```

<echeckPreNoteCreditResponse id="eCheckSale Id" reportGroup="iQ Report Group"
customerId="Customer Id">

  <littleTxnId>Transaction Id</littleTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <message>Response Message</message>
</echeckPreNoteCreditResponse>

```

Example: eCheck Prenotification Credit Response

```

<littleResponse version="11.0" xmlns="http://www.little.com/schema" id="123"
  response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <echeckPreNoteCreditResponse id="AX54321678" reportGroup="RG27"
      customerId="53">
      <littleTxnId>84568456</littleTxnId>
      <orderId>12z58743y1</orderId>
      <response>000</response>
      <responseTime>2017-03-01T10:24:31</responseTime>
      <message>Approved</message>
    </echeckPreNoteCreditResponse>
    <echeckPreNoteCreditResponse id="AX54325432" reportGroup="RG12">
      <littleTxnId>84568457</littleTxnId>
      <orderId>12z58743y7</orderId>
      <response>000</response>
      <responseTime>2017-03-01T10:24:31</responseTime>
      <message>Approved</message>
    </echeckPreNoteCreditResponse>
  </batchResponse>
</littleResponse>

```

3.3.16 eCheck Prenotification Sale Transactions (Batch Only)

You use this non-monetary transaction to verify the consumer's account information prior to submitting an eCheck Sale transaction (also see [eCheck Prenotification](#) on page 47). This transaction type is only supported for US transactions.

3.3.16.1 eCheck Prenotification Sale Request

You must specify the eCheck Prenotification Credit request using the following format:

```
<echeckPreNoteSale id="eCheck PreNote Id" reportGroup="iQ Report Group"
customerID="Customer Id">

    <orderId>Order Id</orderId>
    <orderSource>Order Entry Source</orderSource>
    <billToAddress>
        <echeck>
            <merchantData>
        </echeck>
    </merchantData>
</echeckPreNotesale>
```

Example: eCheck Prenotification Sale Request

```
<littleRequest version="11.0" xmlns="http://www.little.com/schema" id="123"
numBatchRequests="1">
    <authentication>
        <user>userName</user>
        <password>password</password>
    </authentication>
    <batchRequest id="654321" numEcheckPreNoteSale="1" merchantId="100">
        <echeckPreNoteSale id="AX54321678" reportGroup="RG27" customerId="53">
            <orderId>12z58743y1</orderId>
            <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>
        </echeckPreNoteSale>
    </batchRequest>
</littleRequest>
```

```

</billToAddress>
<echeck>
  <accType>Checking</accType>
  <accNum>5186005800001012</accNum>
  <routingNum>000010101</routingNum>
  <checkNum>1104</checkNum>
</echeck>
</echeckPreNoteSale>
</batchRequest>
</littleRequest>

```

3.3.16.2 eCheck Prenotification Sale Response

The eCheck Prenotification Sale response has the following structure:

```

<echeckPreNoteSaleResponse id="eCheckSale Id" reportGroup="iQ Report Group"
  customerId="Customer Id">

  <littleTxnId>Transaction Id</littleTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <message>Response Message</message>
</echeckPreNoteSaleResponse>

```

Example: eCheck Prenotification Sale Response

```

<littleResponse version="11.0" xmlns="http://www.little.com/schema" id="123"
  response="0" message="Valid Format" littleSessionId="987654321">
  <batchResponse id="01234567" littleBatchId="4455667788" merchantId="100">
    <echeckPreNoteSaleResponse id="AX54321678" reportGroup="RG27"
      customerId="53">
      <littleTxnId>84568456</littleTxnId>
      <orderId>12z58743y1</orderId>
      <response>000</response>
      <responseTime>2017-03-01T10:24:31</responseTime>
      <message>Approved</message>
    </echeckPreNoteCreditResponse>
    <echeckPreNoteCreditResponse id="AX54325432" reportGroup="RG12">
      <littleTxnId>84568457</littleTxnId>
      <orderId>12z58743y7</orderId>
      <response>000</response>
      <responseTime>2017-03-01T10:24:31</responseTime>
    </echeckPreNoteCreditResponse>
  </batchResponse>
</littleResponse>

```

```

<message>Approved</message>
</echeckPreNoteSaleResponse>
</batchResponse>
</littleResponse>

```

3.3.17 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.**

3.3.17.1 eCheck Redeposit Request

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

```

<echeckRedeposit id="eCheck Redeposit Id" reportGroup="iQ Report Group"
customerID="Customer Id">

    <littleTxnId>Transaction Id</littleTxnId>
    <echeck> or <echeckToken>
    <merchantData>
        <customIdentifier>
    </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="11.0" xmlns="http://www.little.com/schema">
    <merchantId>81603</merchantId>
    <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>
<merchantData>
  <campaign>New Marketing Campaign</campaign>
</merchantData>
</echeckRedeposit>
</littleOnlineRequest>

```

Example: Batch eCheck Redeposit Request

```

<littleRequest version="11.0" 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>
      <merchantdata>
        <affiliate>ABC Marketing</affiliate>
      </merchantdata>
    </echeckRedeposit>
    <echeckRedeposit reportGroup="001603">
      <littleTxnId>3456456449</littleTxnId>
      <echeck>
        <accType>Checking</accType>
        <accNum>1099999903</accNum>
        <routingNum>114567895</routingNum>
      </echeck>
      <merchantdata>
        <affiliate>ABC Marketing</affiliate>
      </merchantdata>
    </echeckRedeposit>
    <echeckRedeposit reportGroup="001603">
      <littleTxnId>3456557123</littleTxnId>
      <echeck>
        <accType>Savings</accType>
        <accNum>10999999444</accNum>
      </echeck>
    </echeckRedeposit>
  </batchRequest>
</littleRequest>

```

```

<routingNum>114567895</routingNum>
</echeck>
<merchantdata>
  <affiliate>ABC Marketing</affiliate>
</merchantdata>
</echeckRedeposit>
<echeckRedeposit reportGroup="001603">
  <littleTxnId>123456789</littleTxnId>
</echeckRedeposit>
</batchRequest>
</littleRequest>

```

3.3.17.2 eCheck Redeposit Response

The eCheck Redeposit response indicates that we have 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="iQ Report Group"
customerID="Customer Id">

  <littleTxnId>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="11.0" 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>
  </batchResponse>
</littleResponse>

```

```

</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>

```

3.3.18 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`, `zip`, and `phone`.
The value for the `orderSource` element must be one of the following: `telephone`, `ecommerce`, or `recurringtel`.

3.3.18.1 eCheck Sale Request

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

```
<echeckSale id="eCheckSale Id" reportGroup="iQ Report Group"
customerID="Customer Id">
```

```

<orderId>Order Id</orderId>
<verify>true or false</verify>
<amount>Authorization Amount</amount>
<secondaryAmount>Secondary Amount</secondaryAmount>
<orderSource>Order Entry Source</orderSource>
<billToAddress>
<shipToAddress>
<echeck> or <echeckToken>
<customBilling>
<merchantData>
<customIdentifier>
</echeckSale>

```

Example: Batch eCheck Sale Request

```

<littleRequest version="11.0" 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>

```

3.3.18.2 eCheck Sale Response

The eCheck Sale response indicates that we have 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="iQ Report Group"
customerID="Customer Id">

  <littleTxnId>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>
  <tokenResponse> (for Tokenized merchants submitting account data)
</echeckSaleResponse>

```

Example: Batch 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="11.0" 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>
            <response>000</response>
            <responseTime>2011-09-01T10:24:31</responseTime>
            <message>Approved</message>
        </echeckSalesResponse>
        <echeckSalesResponse id="AX54325432" reportGroup="RG12">
            <littleTxnId>84568457</littleTxnId>
            <response>000</response>
            <responseTime>2011-09-01T10:24:31</responseTime>
            <message>Approved</message>
        </echeckSalesResponse>
        <accountUpdater>
            <originalAccountInfo>
                <accType>Checking</accType>
                <accNum>5186005800001012</accNum>
                <routingNum>000010101</routingNum>
            </originalAccountInfo>
            <newAccountInfo>
                <accType>Checking</accType>
                <accNum>5499576040500006</accNum>
                <routingNum>000010102</routingNum>
            </newAccountInfo>
        </accountUpdater>
    </batchResponse>
</littleResponse>

```

3.3.19 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. This transaction type is only supported for US transactions.

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.

3.3.19.1 eCheck Verification Request

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

IMPORTANT: For an eCheckVerification transaction, you must submit the `firstName` and `lastName` elements instead of the `name` element (`middleInitial` is optional). For a corporate account you must include the `companyName` element in addition to the `firstName` and `lastName` elements. In both cases, you also must include the `address`, `city`, `state`, `zip` and `phone` 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.

```
<echeckVerification id="echeckVerification Id" reportGroup="iQ Report Group"
customerId="Customer Id">

    <orderId>Order Id</orderId>
    <amount>Authorization Amount</amount>
    <orderSource>Order Entry Source</orderSource>
    <billToAddress>
        <echeck> or <echeckToken>
        <merchantData>
    </echeckVerification>
```

Example: eCheck Verification Request - Personal Checking

```
<littleRequest version="11.0" 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>
            </billToAddress>
        </echeckVerification>
    </batchRequest>
</littleRequest>
```

```

<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>
<merchantData>
  <campaign>New Campaign</campaign>
</merchantData>
</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="11.0" 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>
      </billToAddress>
    </echeckVerification>
  </batchRequest>
</littleRequest>

```

```

<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>
</titleRequest>8.6

```

3.3.19.2 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="iQ Report
Group" customerId="Customer Id">
  <littleTxnId>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)
  <tokenResponse> (for Tokenized merchants submitting account data)
  <accountUpdater>Account Change Info</accountUpdater>
</echeckSaleResponse>

```

Example: eCheck Verification Response

```
<littleResponse version="11.0" 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>
        <response>000</response>
        <responseTime>2011-09-01T10:24:31</responseTime>
        <message>Approved</message>
    </echeckVerificationResponse>
    <echeckVerificationResponse id="AX54325432" reportGroup="RG12">
        <littleTxnId>84568457</littleTxnId>
        <response>000</response>
        <responseTime>2011-09-01T10:24:31</responseTime>
        <message>Approved</message>
    </echeckVerificationResponse>
    <accountUpdater>
        <originalAccountInfo>
            <accType>Checking</accType>
            <accNum>5186005800001012</accNum>
            <routingNum>000010101</routingNum>
        </originalAccountInfo>
        <newAccountInfo>
            <accType>Checking</accType>
            <accNum>5499576040500006</accNum>
            <routingNum>000010102</routingNum>
        </newAccountInfo>
    </accountUpdater>
</batchResponse>
</littleResponse>

```

3.3.20 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.

3.3.20.1 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="iQ Report Group">
    <littleTxnId>Transaction Id</littleTxnId>
</echeckVoid>
```

Example: eCheck Void Request

```
<littleOnlineRequest version="11.0" 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>
```

3.3.20.2 eCheck Void Response

The eCheck Void response has the following structure.

```
<echeckVoidResponse id="eCheck Void Id" reportGroup="iQ Report Group"
    numDeposits=>"1"

    <littleTxnId>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="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <echeckVoidResponse id="101" reportGroup="001601">
        <littleTxnId>21200000026600</littleTxnId>
        <response>001</response>
        <responseTime>2016-06-17T21:20:50</responseTime>
        <message>Transaction received</message>
        <postDate>2010-06-17</postDate>
    </echeckVoidResponse>
</littleOnlineResponse>
```

3.3.21 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 Vantiv before submitting transactions of this type. In some instances, using a Force Capture transaction can lead to chargebacks and fines.

3.3.21.1 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="iQ Report Group" customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Force Capture Amount</amount>
  <secondaryAmount>Secondary Amount</secondaryAmount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
    [ <card | token> | <paypage> | <mpos> ]
  <customBilling>
    <taxType>payment or fee</taxType>
    <enhancedData>
      <processingInstructions>
        <pos>
          <amexAggregatorData>
          <merchantData>
            <debtRepayment>true or false</debtRepayment>
            <processingType>processingType Enum</processingType>
          </forceCapture>
```

Example: Batch Force Capture Request

```
<littleRequest version="11.0" 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>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>
<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="11.0" xmlns="http://www.little.com/schema">
  <merchantId>123</merchantId>
  <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>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>
    </enhancedData>
  </forceCapture>
</littleOnlineRequest>

```

```

<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>

```

3.3.21.2 Force Capture Response

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

```

<forceCaptureResponse id="Capture Id" reportGroup="iQ Report Group"
customerID="Customer Id">

  <littleTxnId>Transaction Id</littleTxnId>
  <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="11.0" 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>
            <response>000</response>
            <responseTime>2011-09-01T10:24:31</responseTime>
            <message>Approved</message>
        </forceCaptureResponse>
    </batchResponse>
</littleResponse>
```

Example: Online Force Capture Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <forceCaptureResponse 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>
    </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>
    <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>
    </tokenResponse>
</forceCaptureResponse>
```

```

<bin></bin>
</tokenResponse>
</forceCaptureResponse>
```

3.3.22 Fraud Check Transaction

If you wish to retrieve the Advanced Fraud results without introducing a Authorization or Sale transactions, use a Fraud Check transaction, as shown in the example below. Fraud Check transactions are only supported as Online transactions.

3.3.22.1 Fraud Check Request

You must specify the Fraud Check request as follows.

```
<fraudCheck id="Fraud Check Id" reportGroup="iQ Report Group">
    <advancedFraudChecks>
        <billToAddress>
        <shipToAddress>
    </fraudCheck>
```

Example: Fraud Check Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
    merchantId="81601">
    <authentication>
        <user>User Name</user>
        <password>password</password>
    </authentication>
    <fraudCheck id="002" reportGroup="001601">
        <advancedFraudChecks>
            <threatMetrixSessionId>ASDFG-AXXXXAB999</threatMetrixSessionId>
            <customAttribute1>Attribute passed to ThreatMetrix</customAttribute1>
            <customAttribute2>Attribute passed to ThreatMetrix</customAttribute2>
            <customAttribute3>Attribute passed to ThreatMetrix</customAttribute3>
            <customAttribute4>Attribute passed to ThreatMetrix</customAttribute4>
            <customAttribute5>Attribute passed to ThreatMetrix</customAttribute5>
        </advancedFraudChecks>
        <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>
</fraudCheck>
</littleOnlineRequest>

```

3.3.22.2 Fraud Check Response

A Fraud Check response has the following structure.

```

<fraudCheckResponse id="Fraud Check Id" reportGroup="iQ Report Group">
  <littleTxnId>Transaction Id</littleTxnId>
  <response>Response Code</response>
  <message>Response Message</message>
  <responseTime>Date and Time in GMT</responseTime>
  <advancedFraudResults>
</fraudCheckResponse>

```

Example: Fraud Check Response

```

<littleOnlineResponse version="10.3" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <fraudCheckResponse id="002" reportGroup="001601">
    <littleTxnId>82823534116454639</littleTxnId>
    <response>000</response>
    <message>Approved</message>
    <responseTime>2013-11-08T21:36:50</responseTime>
    <advancedFraudResults>
      <deviceReviewStatus>pass</deviceReviewStatus>

```

```

<deviceReputationScore>50</deviceReputationScore>
<triggeredRule>FlashImagesCookiesDisabled</triggeredRule>
</advancedFraudResults>
</fraudCheckResponse>
</littleOnlineResponse>

```

3.3.23 Gift Card Auth Reversal Transactions

The Gift Card Auth Reversal transaction allows you to reverse an authorization against a closed-loop gift card; thereby, removing the funds hold established by the authorization.

3.3.23.1 Gift Card Auth Reversal Request

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

```

<giftCardAuthReversal id="Id" reportGroup="iQ Report Group" customerId="Customer
Id">

    <littleTxnId>Transaction Id from Auth Response</littleTxnId>

    <card>

        <originalRefCode>Auth Code from Auth Response</originalRefCode>
        <originalAmount>Amount from Auth Response</originalAmount>
        <originalTxnTime>txnTime from Auth Response</originalTxnTime>
        <originalSystemTraceId>systemTraceId from Auth Rsp</originalSystemTraceId>
        <originalSequenceNumber>sequenceNumber from Auth Rsp</originalSequenceNumber>
    </giftCardAuthReversal>

```

Example: Online Gift Card Auth Reversal

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
    merchantId="100">

    <authentication>
        <user>User Name</user>
        <password>Password</password>
    </authentication>
    <giftCardAuthReversal id="834262" reportGroup="ABC Division"
        customerId="038945">
        <littleTxnId>82823534116454002</littleTxnId>
        <card>
            <type>GC</type>
            <number>9000000000000001</number>
    </giftCardAuthReversal>
</littleOnlineRequest>

```

```

<PIN>1234</PIN>
</card>
<originalRefCode>123456</originalRefCode>
<originalAmount>40000</originalAmount>
<originalTxnTime>2016-07-25T15:13:43</originalTxnTime>
<originalSystemTraceId>654321</originalSystemTraceId>
<originalSequenceNumber>456789</originalSequenceNumber>
</giftCardAuthReversal>
</littleOnlineRequest>

```

3.3.23.2 Gift Card Auth Reversal Response

A Gift Card Auth Reversal response has the following structure. The response message is identical for Online and Batch transactions except Online includes the <postDate> element.

```

<giftCardAuthReversalResponse id="Load Id" reportGroup="iQ Report
Group" customerId="Customer Id">
    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
    <responseTime>Date and Time in GMT</responseTime>
    <postDate>Date transaction posted</postDate> (Online Only)
    <message>Response Message</message>
    <giftCardResponse>
</giftCardAuthReversalResponse>

```

Example:

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <giftCardAuthReversalResponse id="834262" reportGroup="ABC Division">
        <littleTxnId>9695064321</littleTxnId>
        <response>000</response>
        <responseTime>2017-03-25T15:13:43</responseTime>
        <postDate>2017-03-25</postDate>
        <message>Approved</message>
        <giftCardResponse>
            <txntime>2017-03-25T15:13:38</txntime>
            <refCode>123456</refCode>
            <systemTraceId>123456</systemTraceId>
            <sequenceNumber>123456</sequenceNumber>
            <availableBalance>5000</availableBalance>
        </giftCardResponse>
    </giftCardAuthReversalResponse>
</littleOnlineResponse>

```

```

</giftCardAuthReversalResponse>
</littleOnlineResponse>

```

3.3.24 Gift Card Capture Transactions

The Gift Card Capture transaction allows you to capture (settle) funds previously authorized on a Closed-loop gift card. Unlike the Capture transaction used for a credit card transaction, a Gift Card Capture requires you to provide both the capture amount and the original (authorized) amount. If the transaction is a partial capture, the capture amount will be less than the original amount, with the remainder available for future capture.

3.3.24.1 Gift Card Capture Request

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

```

<giftCardCapture id="Id" reportGroup="iQ Report Group" customerId="Customer Id">
    <littleTxnId>Transaction Id from Auth Response</littleTxnId>
    <captureAmount>Amt of Capture</captureAmount>
    <card>
        <originalRefCode>Auth Code from Auth Response</originalRefCode>
        <originalAmount>Amount from Auth Request</originalAmount>
        <originalTxnTime>txnTime from Auth Response</originalTxnTime>
    </card>
</giftCardCapture>

```

Example: Gift Card Capture Request

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema">
    <merchantId>100</merchantId>
    <authentication>
        <user>User Name</user>
        <password>Password</password>
    </authentication>
    <giftCardCapture id="834262" reportGroup="ABC Division" customerId="038945">
        <littleTxnId>82823534116454002</littleTxnId>
        <captureAmount>40000</captureAmount>
        <card>
            <type>GC</type>
            <number>9000000000000001</number>
        </card>
    </giftCardCapture>
</littleOnlineRequest>

```

```

<originalRefCode>123456</originalRefCode>
<originalAmount>40000</originalAmount>
<originalTxnTime>2016-07-25T15:13:43</originalTxnTime>
</giftCardCapture>
</littleOnlineRequest>

```

3.3.24.2 Gift Card Capture Response

A Gift Card Capture response has the following structure. The response message is identical for Online and Batch transactions except Online includes the <postDate> element.

```

<giftCardCaptureResponse id="Load Id" reportGroup="iQ Report Group"
customerID="Customer Id">

<littleTxnId>Transaction Id</littleTxnId>
<response>Response Code</response>
<responseTime>Date and Time in GMT</responseTime>
<postDate>Date transaction posted</postDate> (Online Only)
<message>Response Message</message>
<accountUpdater>
<fraudResult>
<giftCardResponse>
</giftCardCaptureResponse>

```

Example: Gift Card Capture Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
response="0" message="Valid Format">
<giftCardCaptureResponse id="834262" reportGroup="ABC Division">
<littleTxnId>9695064321</littleTxnId>
<response>000</response>
<responseTime>2017-03-25T15:13:43</responseTime>
<postDate>2017-03-25</postDate>
<message>Approved</message>
<giftCardResponse>
<txnTime>2017-03-25T15:13:38</txnTime>
<refCode>123456</refCode>
<systemTraceId>123456</systemTraceId>
<sequenceNumber>123456</sequenceNumber>
<availableBalance>5000</availableBalance>
</giftCardResponse>
</giftCardCaptureResponse>

```

```
</littleOnlineResponse>
```

3.3.25 Gift Card Credit Transactions

The Gift Card Credit transaction allows you to refund funds previously captured on a Closed-loop gift card. Unlike the Capture transaction used for a credit card transaction, a Gift Card Capture requires you to provide both the capture amount and the original (captured) amount..

3.3.25.1 Gift Card Capture Request

You must specify the Gift Card Capture request as follows. There are two possible structures for this transaction type. You use the structure containing the `littleTxnId` element, when the Vantiv processed the capture transaction. You use the structure containing the `orderId`, when the processor of the capture transaction was not Vantiv. Typically, this occurs when you first migrate your processing to Vantiv. The structures of the requests are identical for either an Online or a Batch submission.

```
<giftCardCapture id="Id" reportGroup="iQ Report Group" customerId="Customer Id">
    <littleTxnId>Transaction Id from Auth Response</littleTxnId>
    <creditAmount>Amt of Credit</captureAmount>
    <card>
</giftCardCapture>
```

or

```
<giftCardCapture id="Id" reportGroup="iQ Report Group" customerId="Customer Id">
    <orderId>Order Id from Capture</littleTxnId>
    <creditAmount>Amt of Credit</captureAmount>
    <orderSource></orderSource>
    <card>
</giftCardCapture>
```

Example: Online Gift Card Credit Transaction (Vantiv Processed Capture)

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema">
    <merchantId>100</merchantId>
    <authentication>
        <user>User Name</user>
        <password>Password</password>
    </authentication>
    <giftCardCredit id="834262" reportGroup="ABC Division"
        customerId="038945">
```

```

<littleTxnId>82823534116454002</littleTxnId>
<creditAmount>40000</creditAmount>
<card>
  <type>GC</type>
  <number>9000000000000001</number>
</card>
</giftCardCredit>
</littleOnlineRequest>

```

Example: Online Gift Card Credit Transaction (Non-Vantiv Processed Capture)

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema">
  <merchantId>100</merchantId>
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <giftCardCredit id="834262" reportGroup="ABC Division"
    customerId="038945">
    <orderId>116454002</orderId>
    <creditAmount>4000</creditAmount>
    <orderSource>ecommerce</orderSource>
    <card>
      <type>GC</type>
      <number>9000000000000001</number>
    </card>
  </giftCardCredit>
</littleOnlineRequest>

```

3.3.25.2 Gift Card Credit Response

A Gift Card Credit response has the following structure. The response message is identical for either request structure, as well as Online and Batch transactions except Online includes the <postDate> element.

```

<giftCardCaptureResponse id="Load Id" reportGroup="iQ Report Group"
  customerId="Customer Id">
  <littleTxnId>Transaction Id</littleTxnId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date transaction posted</postDate> (Online Only)

```

```

<message>Response Message</message>
<fraudResult>
<giftCardResponse>
</giftCardCaptureResponse>

```

Example: Gift Card Credit Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <giftCardCreditResponse id="834262" reportGroup="ABC Division">
    <littleTxnId>9695064321</littleTxnId>
    <response>000</response>
    <responseTime>2017-03-22T15:13:43</responseTime>
    <postDate>2017-03-22</postDate>
    <message>Approved</message>
    <giftCardResponse>
      <txnTime>2017-03-22T12:00:00</txnTime>
      <refCode>003558</refCode>
      <systemTraceId>834528</systemTraceId>
      <sequenceNumber>123456</sequenceNumber>
      <availableBalance>0</availableBalance>
    </giftCardResponse>
  </giftCardCreditResponse>
</littleOnlineResponse>

```

3.3.26 Load Transactions

The Load transaction adds funds to an active Gift Card. The load amount cannot exceed the maximum allowed amount for the Gift Card. If you attempt to load more than the maximum amount, the transaction will be declined with a response Code of 221 - Over Max Balance.

NOTE: You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.

3.3.26.1 Load Request

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

```
<load id="Load Id" reportGroup="iQ Report Group" customerId="Customer Id">
    <orderId>Order Id</orderId>
    <amount>Amount to Load</amount>
    <orderSource>Order Entry Source</orderSource>
    <card>
    </load>
```

Example: Online Load Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
    merchantId="100">
    <authentication>
        <user>User Name</user>
        <password>Password</password>
    </authentication>
    <load id="834262" reportGroup="ABC Division" customerId="038945">
        <orderId>65347567</orderId>
        <amount>40000</amount>
        <orderSource>ecommerce</orderSource>
        <card>
            <type>GC</type>
            <number>9000000000000001</number>
            <cardValidationNum>888</cardValidationNum>
            <pin>1234</pin>
        </card>
    </load>
</littleOnlineRequest>
```

3.3.26.2 Load Response

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

```
<loadResponse id="Load Id" reportGroup="iQ Report Group"
    customerId="Customer Id">
    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
```

```

<responseTime>Date and Time in GMT</responseTime>
<postDate>Date transaction posted</postDate> (Online Only)
<message>Response Message</message>
<fraudResult>
<giftCardResponse>
</loadResponse>

```

Example: Load Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <loadResponse id="834262" reportGroup="ABC Division">
    <littleTxnId>9695064321</littleTxnId>
    <response>000</response>
    <responseTime>2017-03-22T15:13:43</responseTime>
    <postDate>2017-03-22</postDate>
    <message>Approved</message>
    <giftCardResponse>
      <txntime>2017-03-22T15:13:38</txntime>
      <refCode>123456</refCode>
      <systemTraceId>123456</systemTraceId>
      <sequenceNumber>123456</sequenceNumber>
      <availableBalance>5000</availableBalance>
    </giftCardResponse>
  </loadResponse>
</littleOnlineResponse>

```

3.3.27 Load Reversal Transactions (Online Only)

The Load Reversal transaction reverses the operation of a Load transaction, removing the newly loaded amount from the Gift Card. The Load Reversal references the associated Load transaction by means of the littleTxnId element returned in the Load response. You cannot perform a partial Load Reversal. This transaction always reverses the full amount of the referenced Load transaction. This transaction type is available only for Online transactions.

NOTE:	You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.
--------------	---

3.3.27.1 Load Reversal Request

You must structure a Load Reversal request as shown in the following examples.

```
<loadReversal id="Load Reversal Id" reportGroup="iQ Report Group"
customerID="Customer Id">

    <littleTxnId>Transaction Id from Load Response</littleTxnId>
    <card>
        <originalRefCode>Reference Code from Load Response</originalRefCode>
        <originalAmount>Amount from Load Transaction</originalAmount>
        <originalTxnTime>Transaction Time from Load Response</originalTxnTime>
        <originalSystemTraceId>Trace Id from Load Response</originalSystemTraceId>
        <originalSequenceNumber>Seq Num from Load Rsp</originalSequenceNumber>
    </loadReversal>
```

Example: Online Load Reversal Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
merchantId="100">
    <authentication>
        <user>User Name</user>
        <password>Password</password>
    </authentication>
    <loadReversal id="12345" customerId="Customer Id" reportGroup="Load
Reversals">
        <littleTxnId>1234567890123456789</littleTxnId>
        <card>
            <type>GC</type>
            <number>1234102000003558</number>
            <cardValidationNum>888</cardValidationNum>
            <pin>1234</pin>
        </card>
        <originalRefCode>123456</originalRefCode>
        <originalAmount>1900</originalAmount>
        <originalTxnTime>2017-03-21T10:02:46</originalTxnTime>
        <originalSystemTraceId>678901</originalSystemTraceId>
        <originalSequenceNumber>123456</originalSequenceNumber>
    </loadReversal>
</littleOnlineRequest>
```

3.3.27.2 Load Reversal Response

An Load Reversal response has the following structure.

```
<loadReversalResponse id="Load Reversal Id" reportGroup="iQ Report Group" customerId="Customer Id">
    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
    <responseTime>Date and Time in GMT</responseTime>
    <postDate>Date transaction posted</postDate>
    <message>Response Message</message>
    <giftCardResponse>
</loadReversalResponse>
```

Example: Online Load Reversal Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <loadReversalResponse id="834262" reportGroup="ABC Division">
        <littleTxnId>9695064321</littleTxnId>
        <response>000</response>
        <responseTime>2017-03-22T15:13:43</responseTime>
        <postDate>2017-03-22</postDate>
        <message>Approved</message>
        <giftCardResponse>
            <txntime>2017-03-22T12:00:00</txntime>
            <refCode>003558</refCode>
            <systemTraceId>834528</systemTraceId>
            <sequenceNumber>123456</sequenceNumber>
            <availableBalance>0</availableBalance>
        </giftCardResponse>
    </loadReversalResponse>
</littleOnlineResponse>
```

3.3.28 Status Query Transactions (Online Only)

You use a Status Query Transaction to verify the status of an Online transaction submitted within the prior 24 hours. As search criteria, you must submit, at a minimum, the id (the `id` attribute) and transaction type (i.e., authorization, deposit, void, etc.) of the original transaction, but to narrow the search, you can also include the transaction id from the original transaction. This transaction type is available only for Online transactions.

NOTE: You must be specifically enabled to use this transaction type. Please speak to your Implementation Consultant or Relationship Manager for additional information.

3.3.28.1 Query Transaction Request

You must structure your Query request as shown below. The origId and origActionType elements are required.

NOTE: If submitting a query for an eCheck transaction, do not include the account number. Use the origAccountNumber element for Credit/Debit/Gift cards only.

```
<queryTransaction id="GCQueryAuth" reportGroup = "Mer5PM1" customerId="1">
    <origId>id Attribute for Original Transaction</origId>
    <origActionType>Code for type of Original Transaction</origActionType>
    <origLittleTxnId>littleTxnId from Original Transaction</origLittleTxnId>
</queryTransaction>
```

Example: Query Transaction Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
    merchantId="100">
    <authentication>
        <user>User Name</user>
        <password>Password</password>
    </authentication>
    <queryTransaction id="query1" reportGroup="Some RG" customerId="038945">
        <origId>834262</origId>
        <origActionType>A</origActionType>
        <origLittleTxnId>9695061110103040608</origLittleTxnId>
    </queryTransaction>
</littleOnlineRequest>
```

3.3.28.2 Query Transaction Response

The structure of the Query Transaction response message can vary according to the results of the query. The results can include a single or multiple transactions that meet the query criteria, no results, if nothing was found, or a limited response, if a transaction was found, but processing was not complete. The structure of the response message will be as follows:

```

<queryTransactionResponse>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <message>Response Message</message>
  <matchCount>Number of Matches Found</matchCount>
  <results_Max10>
    <queryTransactionUnavailableResponse>
      or
      One or more (10 max) found transaction responses of type specified in the
      queryTransaction
    </queryTransactionUnavailableResponse>
  </results_Max10>
</queryTransactionResponse>

```

Example: Query Transaction Response with One Found Transaction

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <queryTransactionResponse id="GCQueryAuth" reportGroup="Mer5PM1"
    customerId="1">
    <response>150</response>
    <responseTime>2015-04-06T16:40:24</responseTime>
    <message>Original transaction found</message>
    <matchCount>1</matchCount>
    <results_max10>
      <authorizationResponse id="GiftCardAuth" reportGroup="Mer5PM1"
        customerId="1">
        <littleTxnId>82827170811986124</littleTxnId>
        <orderId>150330_GCAuth</orderId>
        <response>000</response>
        <responseTime>2015-04-06T16:40:04</responseTime>
        <postDate>2015-04-06</postDate>
        <message>Approved</message>
        <authCode>111115</authCode>
        <fraudResult>
          <avsResult>30</avsResult>
          <cardValidationResult>M</cardValidationResult>
        </fraudResult>
        <giftCardResponse>
          <availableBalance>125</availableBalance>
        </giftCardResponse>
      </authorizationResponse>
    </results_max10>
  </queryTransactionResponse>
</littleOnlineResponse>

```

```

</queryTransactionResponse>
</littleOnlineResponse>

```

Example: Query Transaction Response with Multiple Found Transactions

NOTE: This response only occurs if you fail to use unique values for the `id` attribute, when submitting transactions.

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
response="0" message="Valid Format">
  <queryTransactionResponse id="GCQueryAuth" reportGroup="Mer5PM1"
  customerId="1">
    <response>150</response>
    <responseTime>2015-04-06T16:40:24</responseTime>
    <message>Original transaction found</message>
    <matchCount>2</matchCount>
    <results_max10>
      <authorizationResponse id="DupeId" reportGroup="Mer5PM1">
        <littleTxnId>82827170811986215</littleTxnId>
        <orderId>150331_DupeAuth2</orderId>
        <response>000</response>
        <responseTime>2015-04-06T16:40:12</responseTime>
        <postDate>2015-04-06</postDate>
        <message>Approved</message>
        <authCode>055858</authCode>
        <fraudResult>
          <avsResult>32</avsResult>
          <cardValidationResult>M</cardValidationResult>
        </fraudResult>
      </authorizationResponse>
      <authorizationResponse id="DupeId" reportGroup="Mer5PM1">
        <littleTxnId>82827170811986207</littleTxnId>
        <orderId>150331_DupeAuth1</orderId>
        <response>000</response>
        <responseTime>2015-04-06T16:40:11</responseTime>
        <postDate>2015-04-06</postDate>
        <message>Approved</message>
        <authCode>111111</authCode>
        <fraudResult>
          <avsResult>00</avsResult>
          <cardValidationResult>M</cardValidationResult>
        </fraudResult>
      </authorizationResponse>
    </results_max10>
  </queryTransactionResponse>
</littleOnlineResponse>

```

```

        </authorizationResponse>
    </results_max10>
</queryTransactionResponse>
</littleOnlineResponse>
```

Example: Query Transaction Response with No Found Transaction

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <queryTransactionResponse id="AuthFromSale" reportGroup="Mer5PM1">
        <response>151</response>
        <responseTime>2015-04-06T16:40:30</responseTime>
        <message>Original transaction not found</message>
        <matchCount>0</matchCount>
        <results_max10></results_max10>
    </queryTransactionResponse>
</littleOnlineResponse>
```

Example: Query Transaction Response with Transaction Found, but not Complete

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <queryTransactionResponse id="someAuth" reportGroup="Mer5PM1">
        <response>152</response>
        <responseTime>2015-04-06T16:40:30</responseTime>
        <message>Original transaction found but response not yet
available</message>
        <matchCount>1</matchCount>
        <results_max10>
            <queryTransactionUnavailableResponse>
                <littleTxnId>82827170811986124</littleTxnId>
                <response>152</response>
                <message>Original transaction found but response not yet
available</message>
            </queryTransactionUnavailableResponse>
        </results_max10>
    </queryTransactionResponse>
</littleOnlineResponse>
```

3.3.29 Refund Reversal Transactions (Online Only)

The Refund Reversal transaction is a Gift Card only transaction that reverses the operation of a Refund transaction on the Gift Card. The Refund Reversal references the associated Credit transaction by means of the `littleTxnId` element returned in the Credit response. You cannot perform a partial Refund Reversal. This transaction always reverses the full amount of the referenced Refund transaction. This transaction type is available only for Online transactions.

NOTE: **You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.**

3.3.29.1 Refund Reversal Request

You must structure a Refund Reversal request as shown in the following examples.

```
<refundReversal id="Refund Reversal Id" reportGroup="iQ Report Group"
customerID="Customer Id">

    <littleTxnId>Transaction Id from CGCredit Response</littleTxnId>
    <card>
        <originalRefCode>Reference Code from CGCredit Response</originalRefCode>
        <originalAmount>Amount from CGCredit Transaction</originalAmount>
        <originalTxnTime>Transaction Time from CGCredit Response</originalTxnTime>
        <originalSystemTraceId>Trace Id from CGCredit Response</originalSystemTraceId>
        <originalSequenceNumber>Seq Num from CGCredit Rsp</originalSequenceNumber>
    </refundReversal>
```

Example: Online Refund Reversal Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
merchantId="100">
    <authentication>
        <user>User Name</user>
        <password>Password</password>
    </authentication>
    <refundReversal id="12345" customerId="Customer Id" reportGroup="Refund
Reversals">
        <littleTxnId>1234567890123456789</littleTxnId>
        <card>
            <type>GC</type>
    </refundReversal>
</littleOnlineRequest>
```

```

<number>1234102000003558</number>
<cardValidationNum>888</cardValidationNum>
<pin>1234</pin>
</card>
<originalRefCode>123456</originalRefCode>
<originalAmount>1900</originalAmount>
<originalTxnTime>2017-03-21T10:02:46</originalTxnTime>
<originalSystemTraceId>678901</originalSystemTraceId>
<originalSequenceNumber>123456</originalSequenceNumber>
</refundReversal>
</littleOnlineRequest>

```

3.3.29.2 Refund Reversal Response

An Refund Reversal response has the following structure.

```

<refundReversalResponse id="Refund Reversal Id" reportGroup="iQ Report
Group" customerId="Customer Id">
    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
    <responseTime>Date and Time in GMT</responseTime>
    <postDate>Date transaction posted</postDate>
    <message>Response Message</message>
    <giftCardResponse>
</refundReversalResponse>

```

Example: Online Refund Reversal Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <refundReversalResponse id="834262" reportGroup="ABC Division">
        <littleTxnId>9695064321</littleTxnId>
        <response>000</response>
        <responseTime>2017-03-22T15:13:43</responseTime>
        <postDate>2017-03-22</postDate>
        <message>Approved</message>
        <giftCardResponse>
            <txntime>2017-03-22T12:00:00</txntime>
            <refCode>003558</refCode>

```

```

<systemTraceId>834528</systemTraceId>
<sequenceNumber>123456</sequenceNumber>
<availableBalance>0</availableBalance>
</giftCardResponse>
</littleOnlineResponse>

```

3.3.30 Register Token Transactions

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

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

3.3.30.1 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 Registration Id.

When you submit the CVV2/CVC2/CID in a registerTokenRequest, the platform encrypts and stores the value on a temporary basis for later use in a tokenized Auth/Sale transaction submitted without the value. This is done to accommodate merchant systems/workflows where the security code is available at the time of token registration, but not at the time of the Auth/Sale. If for some reason you need to change the value of the security code supplied at the time of the token registration, use an updateCardValidationNumOnToken transaction. To use the store value when submitting an Auth/Sale transaction, set the cardValidationNum value to 000.

NOTE: The use of the cardValidationNum element in the registerTokenRequest only applies when you submit an accountNumber element.

For credit cards:

```
<registerTokenRequest id="Id" reportGroup="iQ Report Group">
```

```

<orderId>Order Id</orderId>
<accountNumber>Card Account Number</accountNumber>
<cardValidationNum>CVV2/CVC2/CID</cardValidationNum>
</registerTokenRequest>

```

For eCheck accounts:

```

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

```

For Registration Ids:

```

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

```

For Mobile POS transactions:

```

<registerTokenRequest id="Id" reportGroup="iQ Report Group">
  <orderId>Order Id</orderId>
  <mpos>
    <ksn>Key Serial Number</ksn>
    <formatId>Format of Encrypted Track</formatId>
    <encryptedTrack>Encrypted Track Data</encryptedTrack>
    <track1Status>Track Read Status - 0 or 1</track1Status>
    <track2Status>Track Read Status - 0 or 1</track2Status>
  </mpos>
</registerTokenRequest>

```

For Apple Pay transactions:

```

<registerTokenRequest id="Id" reportGroup="iQ Report Group">
  <orderId>Order Id</orderId>
  <applepay>
    <data>Encrypted Payment Data</data>
  </applepay>
</registerTokenRequest>

```

```

<header>
    <applicationData>Hash of Application Data Property</applicationData>
    <ephemeralPublicKey>Ephemeral Public Key</ephemeralPublicKey>
    <publicKeyHash>Merchant Cert Encoded Public Key</publicKeyHash>
    <transactionId>Transaction Id from Device</transactionId>
</header>
<signature>Signature of Payment and Header Data</signature>
<version>Payment Token Version</version>
</applepay>
</registerTokenRequest>

```

Example: Batch Register Token Request - Credit Card

```

<littleRequest version="11.0" xmlns="http://www.little.com/schema" id="123"
    numBatchRequests="1">
    <authentication>
        <user>userName</user>
        <password>password</password>
    </authentication>
    <batchRequest id="01234567" numTokenRegistrations="1" merchantId="000902">
        <registerTokenRequest id="99999" reportGroup="RG1">
            <orderId>F12345</orderId>
            <accountNumber>4005101001000002</accountNumber>
            <cardValidationNum>999</cardValidationNum>
        </registerTokenRequest>
    </batchRequest>
</littleRequest>

```

Example: Batch Register Token Request - eCheck

```

<littleRequest version="11.0" xmlns="http://www.little.com/schema" id="123"
    numBatchRequests="1">
    <authentication>
        <user>userName</user>
        <password>password</password>
    </authentication>
    <batchRequest id="01234567" numTokenRegistrations="1" merchantId="000902">
        <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="11.0" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest id="01234567" numTokenRegistrations="1" merchantId="000902">
    <registerTokenRequest id="99999" reportGroup="RG1">
      <orderId>F12345</orderId>
      <paypageRegistrationId>12345678901234567</paypageRegistrationId>
    </registerTokenRequest>
  </batchRequest>
</littleRequest>

```

3.3.30.2 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>Transaction Id</littleTxnId>
  <littleToken>Token</littleToken>
  <bin>BIN</bin>
  <type>Method of Payment</type>
  <response>Response Code</response>
  <message>Response Message</message>
  <responseTime>Response Time</responseTime>
</registerTokenResponse>

```

Register Token response for eChecks:

```
<registerTokenResponse id="99999" reportGroup="RG1">
    <littleTxnId>Transaction Id</littleTxnId>
    <littleToken>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>
```

Register Token response for ApplePay:

```
<registerTokenResponse id="99999" reportGroup="RG1">
    <littleTxnId>Transaction Id</littleTxnId>
    <littleToken>Token</littleToken>
    <type>Method of Payment</type>
    <response>Response Code</response>
    <responseTime>Response Time</responseTime>
    <message>Response Message</message>
    <applepayResponse>
        <applicationExpirationDate>App PAN Exp Date</applicationExpirationDate>
        <currencyCode>Currency Code</currencyCode>
        <transactionAmount>Amount of Transaction</transactionAmount>
        <cardholderName>Name of cardholder</cardholderName>
        <deviceManufacturerIdentifier>Device Mfr Id</deviceManufacturerIdentifier>
        <paymentDataType>Type of Payment Data</paymentDataType>
        <onlinePaymentCryptogram>Payment Cryptogram</onlinePaymentCryptogram>
        <eciIndicator>eCommerce Indicator</eciIndicator>
    </applepayResponse>
</registerTokenResponse>
```

Example: Batch Register Token Response - Credit Card

```
<littleResponse version="11.0" 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>
            <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>
            <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>

```

3.3.31 RFR Transactions (Batch Only)

An RFR (Request For Response) transaction enables you to request a response file for a previously submitted Batch. 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.

3.3.31.1 RFR Request

You must specify the RFR request as follows.

```
<RFRequest>
```

```
<littleSessionId | accNum>
</RFRequest>
```

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="11.0" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="0">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <RFRequest>
    <littleSessionId>7766554321</littleSessionId>
  </RFRequest>
</littleRequest>
```

Example: RFR Request for an Account Updater File

```
<littleRequest version="11.0" xmlns="http://www.little.com/schema" id="123"
  numBatchRequests="0">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <RFRequest>
    <merchantId>100</merchantId>
    <postDate>2010-01-15</postDate>
  </RFRequest>
</littleRequest>
```

3.3.31.2 RFR Response

When using an RFR request to obtain the response file for a payment transaction Batch, 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="11.0" xmlns="http://www.little.com/schema">
  <RFRResponse response="1" message="The account update file is not ready
yet. Please try again later.">
  </RFRResponse>
</littleResponse>
```

3.3.32 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, except for American Express transactions. For American Express, a failure to match the security code (CID) results in a declined transaction with the Response Reason Code of 352 - Decline CVV2/CID Fail.

3.3.32.1 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.

NOTE: When you submit the CVV2/CVC2/CID in a registerTokenRequest, the platform encrypts and stores the value on a temporary basis for later use in a tokenized Auth/Sale transaction submitted without the value. To use the store value when submitting an Auth/Sale transaction, set the cardValidationNum value to 000.

```
<sale id="Sale Id" reportGroup="iQ Report Group" customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Authorization Amount</amount>
  <secondaryAmount>Secondary Amount</secondaryAmount>
  <orderSource>Order Entry Source</orderSource>
  <customerInfo>
    <billToAddress>
      <shipToAddress>
```

```
[<card>|<paypage>|<token>|<paypal>|<mpos>|<applepay>|
<sepaDirectDebit>|<ideal>] (Choice)
<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> (for Recycling Engine merchants)
<fraudFilterOverride>
<recurringRequest> (for Recurring Engine merchants)
<debtRepayment>true or false</debtRepayment>
<advancedFraudChecks>
<wallet>
<processingType>processingType Enum</processingType>
<originalNetworkTransactionId>Network Txn Value</originalNetworkTransactionId>
<originalTransactionAmount>Amount from Orig Txn</originalTransactionAmount>
</sale>
```

Example: Batch Sale Request

```
<littleRequest version="11.0" 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>
</titleRequest>

```

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="11.0" xmlns="http://www.little.com/schema">
  <merchantId>100</merchantId>
  <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&amp;Co Test</descriptor>
</customBilling>
<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>2011-08-14</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>

```

Example: Online Sale Request - SEPA Direct Debit

```

<littleOnlineRequest version="10.5" xmlns="http://www.little.com/schema">
  <merchantId>100</merchantId>
  <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>
<sepaDirectDebit>
<mandateProvider>Vantiv</mandateProvider>
<sequenceType>OneTime</sequenceType>
<iban>DE89370400440532013000</iban>
<preferredLanguage>DE</preferredLanguage>
</sepaDirectDebit>
</sale>
</littleOnlineRequest>

```

3.3.32.2 Sale Response

The Sale response message is identical for Online and Batch transactions except Online includes the postDate element. The Sale response has the following structure:

```

<saleResponse id="Sale Id" reportGroup="iQ Report Group" customerId="Customer
Id">

<littleTxnId>Transaction Id</littleTxnId>
<response>Response Code</response>
<orderId>Order Id</orderId>
<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 feature is enabled)
<recurringResponse> (for Recurring Engine merchants)
<giftCardResponse> (included if Gift Card is Method of Payment)
<applepayResponse> (included if an ApplePay transaction)
<cardSuffix>Card Last 4</cardSuffix> (included for ApplePay using VI or MC)
<androidpayResponse>
<sepaDirectDebitResponse>
<networkTransactionId>Txn ID returned from network</networkTransactionId>
</saleResponse>

```

Example: Batch Sale Response

```

<littleResponse version="11.0" 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>
      <response>000</response>
      <orderId>12z58743y1</orderId>
      <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>
      <response>000</response>
      <orderId>12z58743y7</orderId>
      <responseTime>2011-09-01T10:24:31</responseTime>
      <message>Approved</message>
    </saleResponse>
  </batchResponse>
</littleResponse>

```

```

<authCode>123456</authCode>
<fraudResult>
  <avsResult>00</avsResult>
  <authenticationResult>2</authenticationResult>
</fraudResult>
</saleResponse>
</batchResponse>
</littleResponse>

```

Example: Online Sale Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <saleResponse id="1" reportGroup="ABC Division" customerId="038945">
    <littleTxnId>1100030055</littleTxnId>
    <response>000</response>
    <orderId>23423434</orderId>
    <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>
  <response>000</response>
  <orderId>F12345</orderId>
  <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="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <saleResponse id="1" reportGroup="ABC Division" customerId="038945">
    <littleTxnId>1100030055</littleTxnId>
    <response>000</response>
    <orderId>23423434</orderId>
    <responseTime>2011-07-11T14:48:46</responseTime>
    <postDate>2011-07-11</postDate>
    <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>

```

Example: Batch Sales Response with Recurring Info

The following is an example of the Recurring Response file produced daily to provide information about the transactions submitted by the Recurring Engine. This file is delivered via sFTP.

```
<littleResponse version="10.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format" littleSessionId="82912082263408653">
    <batchResponse littleBatchId="82912082263408661" merchantId="101">
        <saleResponse reportGroup="Default Report Group">
            <littleTxnId>82912082263409610</littleTxnId>
            <response>000</response>
            <orderId>recurring_appr1</orderId>
            <responseTime>2014-01-30T20:15:51</responseTime>
            <message>Approved</message>
            <authCode>11111</authCode>
            <fraudResult>
                <avsResult>01</avsResult>
            </fraudResult>
            <recurringResponse>
                <subscriptionId>82912081866997773</subscriptionId>
                <responseCode>473</responseCode>
                <responseMessage>Scheduled recurring payment
processed</responseMessage>
                <recurringTxnId>211014241510</recurringTxnId>
            </recurringResponse>
        </saleResponse>
        <saleResponse reportGroup="Default Report Group">
            <littleTxnId>82912082263410311</littleTxnId>
            <response>000</response>
            <orderId>recurring_appr2</orderId>
            <responseTime>2014-01-30T20:15:55</responseTime>
            <message>Approved</message>
            <authCode>123457</authCode>
            <fraudResult>
                <avsResult>00</avsResult>
            </fraudResult>
            <recurringResponse>
                <subscriptionId>82912081866997799</subscriptionId>
                <responseCode>473</responseCode>
                <responseMessage>Scheduled recurring payment
processed</responseMessage>
```

```
<recurringTxnId>211014245016</recurringTxnId>
</recurringResponse>
</saleResponse>
<saleResponse reportGroup="Default Report Group">
<littleTxnId>82912082263410337</littleTxnId>
<response>110</response>
<orderId>recurring_decline1</orderId>
<responseTime>2014-01-30T20:15:56</responseTime>
<message>Insufficient Funds</message>
<fraudResult>
<avssResult>34</avssResult>
</fraudResult>
<recycling>
<recycleEngineActive>true</recycleEngineActive>
</recycling>
<recurringResponse>
<subscriptionId>82912081866997807</subscriptionId>
<responseCode>473</responseCode>
<responseMessage>Scheduled recurring payment
processed</responseMessage>
<recurringTxnId>211014245115</recurringTxnId>
</recurringResponse>
</saleResponse>
<saleResponse reportGroup="Default Report Group">
<littleTxnId>82912082263410378</littleTxnId>
<response>110</response>
<orderId>recurring_decline3</orderId>
<responseTime>2014-01-30T20:15:56</responseTime>
<message>Insufficient Funds</message>
<fraudResult>
<avssResult>34</avssResult>
</fraudResult>
<recycling>
<recycleEngineActive>true</recycleEngineActive>
</recycling>
<recurringResponse>
<subscriptionId>82912081866997807</subscriptionId>
<responseCode>473</responseCode>
<responseMessage>Scheduled recurring payment
processed</responseMessage>
<recurringTxnId>211014245313</recurringTxnId>
</recurringResponse>
</saleResponse>
```

```
</batchResponse>
</littleResponse>
```

3.3.33 Unload Transactions

The Unload transaction removes funds from an active Gift Card. The unload amount cannot exceed the available balance on the Gift Card. If you attempt to unload more than the available balance, the transaction will be declined with a response Code of 209 - Invalid Amount.

NOTE: You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.

3.3.33.1 Unload Request

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

```
<unload id="Unload Id" reportGroup="iQ Report Group" customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Amount to Load</amount>
  <orderSource>Order Entry Source</orderSource>
  <card>
</unload>
```

Example: Online Unload Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <unload id="834262" reportGroup="ABC Division" customerId="038945">
    <orderId>65347567</orderId>
    <amount>40000</amount>
    <orderSource>ecommerce</orderSource>
    <card>
      <type>GC</type>
      <number>9000000000000001</number>
      <pin>1234</pin>
    </card>
  </unload>
</littleOnlineRequest>
```

```

</card>
</onload>
</littleOnlineRequest>
```

3.3.33.2 Unload Response

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

```

<unloadResponse id="Unload Id" reportGroup="iQ Report Group"
customerID="Customer ID">

    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
    <responseTime>Date and Time in GMT</responseTime>
    <postDate>Date transaction posted</postDate> (Online Only)
    <message>Response Message</message>
    <fraudResult>
    <giftCardResponse>
</unloadResponse>
```

Example: Unload Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
response="0" message="Valid Format">
    <unloadResponse id="834262" reportGroup="ABC Division">
        <littleTxnId>9695064321</littleTxnId>
        <response>000</response>
        <responseTime>2017-03-22T15:13:43</responseTime>
        <postDate>2017-03-22</postDate>
        <message>Approved</message>
        <giftCardResponse>
            <txnPTime>2017-03-22T12:00:00</txnPTime>
            <refCode>003558</refCode>
            <systemTraceId>834528</systemTraceId>
            <sequenceNumber>123456</sequenceNumber>
            <availableBalance>0</availableBalance>
        </giftCardResponse>
    </unloadResponse>
</littleOnlineResponse>
```

3.3.34 Unload Reversal Transactions (Online Only)

The Unload Reversal transaction reverses the operation of a Unload transaction, returning the value removed from the Gift Card by the Unload transaction. The Unload Reversal references the associated Unload transaction by means of the `littleTxnId` element returned in the Unload response. You cannot perform a partial Unload Reversal. This transaction always reverses the full amount of the referenced Unload transaction. This transaction type is available only for Online transactions.

NOTE: You must be enabled for (Closed Loop) Gift Card transactions to use this transaction type. Please consult your Relationship Manager for additional information about Gift Card transactions.

3.3.34.1 Unload Reversal Request

You must structure a Unload Reversal request as shown in the following examples.

```
<unloadReversal id="Unload Reversal Id" reportGroup="iQ Report Group"
customerID="Customer Id">

    <littleTxnId>Transaction Id from CGCredit Response</littleTxnId>

    <card>

        <originalRefCode>Reference Code from Unload Response</originalRefCode>
        <originalAmount>Amount from Unload Transaction</originalAmount>
        <originalTxnTime>Transaction Time from Unload Response</originalTxnTime>
        <originalSystemTraceId>Trace Id from Unload Response</originalSystemTraceId>
        <originalSequenceNumber>Seq Num from Unload Rsp</originalSequenceNumber>

    </unloadReversal>
```

Example: Online Unload Reversal Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
merchantId="100">
    <authentication>
        <user>User Name</user>
        <password>Password</password>
    </authentication>
    <unloadReversal id="12345" customerId="Customer Id" reportGroup="Unload
Reversals">
        <littleTxnId>1234567890123456789</littleTxnId>
        <card>
            <type>GC</type>
            <number>1234102000003558</number>
            <cardValidationNum>888</cardValidationNum>
    </unloadReversal>
</littleOnlineRequest>
```

```

<pin>1234</pin>
</card>
<originalRefCode>123456</originalRefCode>
<originalAmount>1900</originalAmount>
<originalTxnTime>2017-03-21T10:02:46</originalTxnTime>
<originalSystemTraceId>678901</originalSystemTraceId>
<originalSequenceNumber>123456</originalSequenceNumber>
</unloadReversal>
</littleOnlineRequest>

```

3.3.34.2 Unload Reversal Response

An Unload Reversal response has the following structure.

```

<unloadReversalResponse id="Unload Reversal Id" reportGroup="iQ Report
Group" customerId="Customer Id">

    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
    <responseTime>Date and Time in GMT</responseTime>
    <postDate>Date transaction posted</postDate>
    <message>Response Message</message>
    <giftCardResponse>
</unloadReversalResponse>

```

Example: Online Unload Reversal Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <unloadReversalResponse id="834262" reportGroup="ABC Division">
        <littleTxnId>9695064321</littleTxnId>
        <response>000</response>
        <responseTime>2017-03-22T15:13:43</responseTime>
        <postDate>2017-03-22</postDate>
        <message>Approved</message>
        <giftCardResponse>
            <txntime>2017-03-22T12:00:00</txntime>
            <refCode>003558</refCode>
            <systemTraceId>834528</systemTraceId>
            <sequenceNumber>123456</sequenceNumber>
            <availableBalance>0</availableBalance>

```

```

</giftCardResponse>
</unloadReversalResponse>
</littleOnlineResponse>

```

3.3.35 Update Plan Transactions

You use the Update Plan transaction to activate/deactivate Plans associated with recurring payments. When you deactivate a Plan, by setting the `active` flag to `false`, you can no longer reference that Plan for use with subscriptions. Existing subscriptions already using the deactivated Plan will continue to use the Plan until the subscription is either modified or completed. You can also reactivate a deactivated Plan by updating the Plan and setting the `active` flag to `true`.

3.3.35.1 Update Plan Request

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

```

<updatePlan>
  <planCode>Plan Reference Code</planCode>
  <active>true or false</active>
</updatePlan>

```

Example: Online Update Plan Request

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema">
  <merchantId="100">
    <authentication>
      <user>User Name</user>
      <password>password</password>
    </authentication>
    <updatePlan>
      <planCode>Reference_Code</planCode>
      <active>false</active>
    </updatePlan>
  </littleOnlineRequest>

```

3.3.35.2 Update Plan Response

An Update Plan response has the following structure. The response message is identical for Online and Batch transactions.

```
<updatePlanResponse>
```

```

<littleTxnId>Transaction Id</littleTxnId>
<response>Response Code</response>
<message>Response Message</message>
<responseTime>Date and Time in GMT</responseTime>
<planCode>Plan Reference Code</subscriptionId>
</updatePlanResponse>

```

Example: Online Update Plan Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <updatePlanResponse>
    <littleTxnId>1100030055</littleTxnId>
    <response>000</response>
    <message>Approved</message>
    <responseTime>2016-07-11T14:48:46</responseTime>
    <planCode>Reference_Code</planCode>
  </updatePlanResponse>
</littleOnlineResponse>

```

3.3.36 Update Subscription Transactions

The Update Subscription transaction allows you to change certain subscription information associated with a recurring payment. Using this transaction type you can change the plan, card, billing information, and/or billing date. You can also create, update, or delete a Discount and/or an Add On.

NOTE: You can include multiple create, update, and delete Discounts and Add On operations in a single updateSubscription transaction.

3.3.36.1 Update Subscription Request

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

```

<updateSubscription>
  <subscriptionId>Subscription Id</subscriptionId>
  <planCode>Plan Reference Code</subscriptionId>
  <billToAddress>
    [ <card> | <paypage> | <token> ]

```

```

<billingDate>New Billing Date</billingDate>
<createDiscount>
<updateDiscount>
<deleteDiscount>
<createAddOn>
<updateAddOn>
<deleteAddOn>
</updateSubscription

```

Example: Online Update Subscription Request

```

<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>password</password>
  </authentication>
  <updateSubscription>
    <subscriptionId>1234</subscriptionId>
    <planCode>Gold_Monthly</planCode>
    <billToAddress>
      <name>John Smith</name>
      <addressLine1>100 Main St</addressLine1>
      <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>
    <deleteDiscount>
      <discountCode>1GBExtra</discountCode>
    </deleteDiscount>
    <createAddOn>
      <addOnCode>1WF</addOnCode>
      <name>One_GB_Extra</name>
    
```

```

<amount>500</amount>
<startDate>2016-09-15</startDate>
<endDate>2016-09-22</endDate>
</createAddOn>
</updateSubscription>
</littleOnlineRequest>

```

3.3.36.2 Update Subscription Response

An Update Subscription response has the following structure. The response message is identical for Online and Batch transactions.

```

<updateSubscriptionResponse>
  <littleTxnId>Transaction Id</littleTxnId>
  <response>Response Code</response>
  <message>Response Message</message>
  <responseTime>Date and Time in GMT</responseTime>
  <subscriptionId>ID of Subscription Canceled</subscriptionId>
  <tokenResponse> (For Tokenized Merchants submitting Card/Paypage)
</cancelSubscriptionResponse>

```

Example: Online Update Subscription Response

```

<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format">
  <updateSubscriptionResponse>
    <littleTxnId>1100030055</littleTxnId>
    <response>001</response>
    <message>Transaction received</message>
    <responseTime>2016-07-11T14:48:46</responseTime>
    <subscriptionId>123457</subscriptionId>
  </updateSubscriptionResponse>
</littleOnlineResponse>

```

3.3.37 Update Card Validation Number Transactions

When you submit the CVV2/CVC2/CID in a registerTokenRequest, the platform encrypts and stores the value on a temporary basis for later use in a tokenized Auth/Sale transaction submitted without the value. This is done to accommodate merchant systems/workflows where

the security code is available at the time of token registration, but not at the time of the Auth/Sale. If for some reason you need to change the value of the security code supplied at the time of the token registration, use an updateCardValidationNumOnToken transaction.

NOTE: You should only use this transaction type if you had previously submitted the account number and security code in a registerTokenRequest transaction and now need to change the CVV2/CVC2/CID value.

3.3.37.1 Update Card Validation Number Request

The updateCardValidationNumOnToken transaction has the following structure:

```
<updateCardValidationNumOnToken id = "Update Id" customerId="Customer Id"
reportGroup="iQ Report Group">
    <orderId>Order Id</orderId>
    <littleToken>Token</littleToken>
    <cardValidationNum>CVV2/CVC2/CID Value</cardValidationNum>
</updateCardValidationNumOnToken>
```

Example: Online Update Card Validation Number Request

```
<littleOnlineRequest version="11.0" xmlns="http://www.little.com/schema"
merchantId="100">
    <updateCardValidationNumOnToken id="99999" customerId="444"
reportGroup="RG1">
        <orderId>F12345</orderId>
        <littleToken>1111000101039449</littleToken>
        <cardValidationNum>987</cardValidationNum>
    </updateCardValidationNumOnToken>
</littleOnlineRequest>
```

3.3.37.2 Update Card Validation Number Response

The updateCardValidationNumOnTokenResponse transaction has the following structure:

```
<updateCardValidationNumOnTokenResponse id="Update Id" reportGroup="iQ Report
Group">
    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
    <message>Response Message</message>
    <responseTime>Date and Time in GMT</responseTime>
```

```
</updateCardValidationNumOnTokenResponse>
```

Example: Online Update Card Validation Number Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <updateCardValidationNumOnTokenResponse id="99999" customerId="444"
        reportGroup="RG1">
        <littleTxnId>21122700</littleTxnId>
        <response>803</response>
        <message>Card Validation Number Updated</message>
        <responseTime>2012-08-09T17:21:51</responseTime>
    </updateCardValidationNumOnTokenResponse>
</littleOnlineResponse>
```

3.3.38 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. Also, if you use Recycling Engine, you can use the void transaction to halt the recycling of a sale transaction. In this case the response may include the recycling element. (see [Using Void to Halt Recycling Engine](#) on page 85).

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 220.)**

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
Sale Transactions	Credit Transactions
Credit Transactions	Sale Transactions
Force Capture Transactions	Credit Transactions

3.3.38.1 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="iQ Report Group">
    <littleTxnId>Transaction Id</littleTxnId>
    <processingInstructions>
</void>
```

Example: Online Void Request

```
<littleOnlineRequest version="11.0" 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>
```

3.3.38.2 Void Response

The Void response has the following structure.

```
<voidResponse id="Void Id" reportGroup="iQ Report Group">
    <littleTxnId>Transaction Id</littleTxnId>
    <response>Response Code</response>
    <responseTime>Date and Time in GMT</responseTime>
    <postDate>Date of Posting</postDate>
    <message>Response Message</message>
    <recycling> (May be included if halting recycling.)
</voidResponse>
```

Example: Online Void Response

```
<littleOnlineResponse version="11.0" xmlns="http://www.little.com/schema"
    response="0" message="Valid Format">
    <voidResponse id="1" reportGroup="Void Division">
        <littleTxnId>1100026202</littleTxnId>
        <response>001</response>
```

```
<responseTime>2011-07-16T19:43:38</responseTime>
<postDate>2011-07-16</postDate>
<message>Transaction received</message>
</voidResponse>
</littleOnlineResponse>
```

Example: Online Void Response with Recycling Element

When you use a Void transaction to halt recycling, the response may include the recycling element. (see [Using Void to Halt Recycling Engine](#) on page 85).

```
<littleOnlineResponse version="10.0" xmlns="http://www.little.com/schema">
  <response>0</response>
  <voidResponse id="1" reportGroup="Void Division">
    <littleTxnId>1100026202333456789</littleTxnId>
    <response>001</response>
    <responseTime>2011-07-16T19:43:38</responseTime>
    <postDate>2011-07-16</postDate>
    <message>Transaction received</message>
    <recycling>
      <creditLittleTxnId>1234567890123456789</creditLittleTxnId>
    </recycling>
  </voidResponse>
</littleOnlineResponse>
```

LITLEXML ELEMENTS

This chapter provides definitions for the elements and attributes used in LittleXML. This information is intended to be used in combination with the various LittleXML schema files to assist you as you build the code necessary to submit transactions to our 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 LittleXML requests and responses using these elements, as well as XML examples, please refer to [Chapter 3, "LittleXML 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 accNum element, the number must be greater than or equal to 4 characters for the transaction to succeed.

Parent Elements:

echeck, newAccountInfo, originalAccountInfo, accountInfo

Attributes:

None

Child Elements:

None

4.2 accountInfo

The accountInfo element is a required child of the submerchantCredit, and submerchantDebit transactions. It contains child elements used to provide details concerning the Sub-merchant account.

Parent Elements:

[submerchantCredit](#), [submerchantDebit](#), [vendorCredit](#), [vendorDebit](#)

Attributes:

None

Child Elements:

Required: [accType](#), [accNum](#), [routingNum](#)

Optional: [ccdPaymentInformation](#)

NOTE: Although shown as an optional element in the schema, the `checkNum` element should not be used as a child of `accountInfo`.

Example: accountInfo Structure

```
<accountInfo>
    <accType>Account Type Abbreviation</accType>
    <accNum>Account Number</accNum>
    <routingNum>Routing Number</routingNum>
    <ccdPaymentInformation>Payment Description</ccdPaymentInformation>
</accountInfo>
```

4.3 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](#)

4.4 accountNumber

The `accountNumber` element is a required child of the `registerTokenRequest` element defining the account number for which you are requesting a token. It is also an optional child of the `virtualGiftCardResponse` element, where it defines the account number of the requested Virtual Gift Card.

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

Parent Elements:

[registerTokenRequest](#), [virtualGiftCardResponse](#)

Attributes:

None

Child Elements:

None

4.5 accountNumberLength

The accountNumberLength element is a required child of the virtualGiftCard element defining the requested length of the virtual Gift Card number you are requesting.

NOTE: In an early iteration of schema V8.22 issued in September of 2013 this element was defined as an optional child of the virtualGiftCard element. If you coded to the earlier version, be aware that this element is now required. If you do not include this element, the transaction will fail XML validation.

Type = Integer; Allowed Values between 13 and 25 inclusive.

IMPORTANT: Although the schema defines the allowed values as any integer between 13 and 25 inclusive, at this time you can only use a value of either 16 or 19.

Parent Elements:

[virtualGiftCard](#)

Attributes:

None

Child Elements:

None

4.6 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	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (all Required)

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

4.7 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>
```

4.8 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 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

IMPORTANT: When using Account Updater (any variation), you must always code to receive the extendedCardResponse element and its children. Vantiv always returns this information whenever applicable regardless of whether you receive other account updater information in the transaction response message.

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>
</accountUpdater>
```

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>
```

4.9 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	Yes	The response returns the same value submitted in the accountUpdate transaction. minLength = 1 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](#)

4.10 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, accountInfo

Attributes:

None

Child Elements:

None

4.11 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 in an authReversal transaction, the information will be forwarded to MasterCard as part of the MasterCard eCommerce Fraud Alert program.

When you include this optional element in an credit transaction, the system uses the information to track potentially fraudulent transactions for future analysis.

Parent Elements:

authReversal, credit

Attributes:

None

Child Elements:

None

4.12 activate

The `activate` element is the parent element for the transaction type that activates a Gift Card.

Parent Elements:

`titleOnlineRequest`, `batchRequest`

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (Required)

`orderId`, `amount`, `orderSource`, `card`, `virtualGiftCard`

4.13 activateResponse

The `activateResponse` element is the parent element for information returned to you in response to an `activate` transaction. You can use this element in either Online or Batch transactions.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#), [giftCardResponse](#)

Optional: [postDate](#), [fraudResult](#), [giftCardResponse](#), [virtualGiftCardResponse](#)

4.14 activateReversal

The `activateReversal` element is the parent element for the transaction type that reverses the activation of a Gift Card.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (Required)

[card](#), [originalRefCode](#), [originalAmount](#), [originalTxnTime](#), [originalSystemTraceId](#), [originalSequenceNumber](#)

Child Elements: (Optional)

[littleTxnId](#), [virtualGiftCardBin](#)

4.15 activateReversalResponse

The `activateReversalResponse` element is the parent element for information returned to you in response to an `activateReversal` transaction. You can use this element in either Online or Batch transactions.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#), [giftCardResponse](#)

Optional: [postDate](#)

4.16 active

The `active` element is an optional child of both the `createPlan` and `updatePlan` elements. You use this flag to activate/deactivate a Plan. When you deactivate a Plan, you can no longer reference that Plan for use with subscriptions. Existing subscriptions making use of the deactivated Plan will continue to use the Plan until either modified or completed. You can also reactivate a deactivated Plan by updating the Plan and setting the `active` flag to true.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[createPlan](#), [updatePlan](#)

Attributes:

None

Child Elements:

None

4.17 addOnCode

The addOnCode element is the identifier of a defined add on charge to a subscription. You use this element when creating, updating, and deleting an Add On applied to a subscription.

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

Parent Elements:

[createAddOn](#), [updateAddOn](#), [deleteAddOn](#)

Attributes:

None

Child Elements:

None

4.18 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

4.19 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

4.20 advancedAVSResult

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 838.

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 Relationship Manager for additional information.

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

Parent Elements:

[fraudResult](#)

Attributes:

None

Child Elements:

None

4.21 advancedFraudChecks

The `advancedFraudChecks` element is an optional child of both the `authorization` and `sale` elements and a required child of the `fraudCheck` element.

Parent Elements:

[authorization](#), [sale](#), [fraudCheck](#)

Attributes:

None

Child Elements:

[threatMetrixSessionId](#), [customAttribute1](#) through [customAttribute5](#)

Example: advancedFraudChecks Structure

```
<advancedFraudChecks>
    <threatMetrixSessionId>Session Id from Threat Metrix</threatMetrixSessionId>
    <customAttribute1>Some attribute passed to ThreatMetrix</customAttribute1>
    <customAttribute2>Some attribute passed to ThreatMetrix</customAttribute2>
    <customAttribute3>Some attribute passed to ThreatMetrix</customAttribute3>
    <customAttribute4>Some attribute passed to ThreatMetrix</customAttribute4>
    <customAttribute5>Some attribute passed to ThreatMetrix</customAttribute5>
</advancedFraudChecks>
```

4.22 advancedFraudResults

The `advancedFraudResults` element is an optional child of both the `fraudResult` and the `fraudCheckResponse` elements. Child elements return the results of advanced fraud checks performed by Threat Metrix, as well as a list of the rules triggered from the ThreatMetrix (merchant) Policy.

Parent Elements:

`fraudResult`, `fraudCheckResponse`

Attributes:

None

Child Elements: (Required)

`deviceReviewStatus`, `deviceReputationScore`, `triggeredRule`

Example: advancedFraudChecks Structure

```
<advancedFraudResults>
    <deviceReviewStatus>pass, fail, review, or unavailable</deviceReviewStatus>
    <deviceReputationScore>Score Returned from ThreatMetix</deviceReputationScore>
    <triggeredRule>Triggered Rule #1</triggeredRule>
    .
    .
    .
    <triggeredRule>Triggered Rule #N</triggeredRule>
</advancedFraudResults>
```

4.23 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.

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

Parent Elements:

[merchantData](#)

Attributes:

None

Child Elements:

None

4.24 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.

4.25 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.

NOTE: If you settle in a currency other than US or Canada, you cannot use partial authorizations.

When submitting transactions for private label gift cards (card type of GC), the use of partial authorizations is determined by a setting in the Merchant Profile (on or off globally). This flag has no effect.

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.

4.26 amexAggregatorData

The amexAggregatorData element defines Amex Aggregator specific information in the LittleXML. 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>
```

4.27 amount

The amount element is a child of several elements. When used in a payment transaction this element defines the amount of the transaction. When amount is a child of the subscription element, it defines the amount of the recurring payment. As a child of the activate, load, or unload transactions, amount defines the initial value of a newly activated gift Card, the amount loaded onto a reloadable Gift Card, or the amount unloaded from a Gift Card, respectively. When used in an Instruction-Based Dynamic Payout transaction type, it specifies the amount of funds to transfer. Supply the value in cents without a decimal point. For example, a value of 1995 signifies \$19.95.

NOTE: For the following currencies, the value you submit is the amount without implied decimal: Burundian Franc, Chilean Peso, Comoro Franc, Djibouti Franc, Guinea Franc, Iceland Krona, Japanese Yen, South Korean Won, Vanuatu Vatu, Paraguayan Guarani, Rawanda Franc, Vietnamese Dong, Uganda Shilling, CFA Franc BEAC, CFA Franc BCEAO, and CFP Franc.
For example, if the currency is Japanese Yen, entering 2000 means 2000 Yen, NOT 20.00 Yen.

Type = Integer; totalDigits = 12

Parent Elements:

The amount element is a required child of each of the following Parent Elements: [activate](#), [authorization](#), [credit](#) (required if original transaction was not processed by Vantiv), [captureGivenAuth](#), [echeckCredit](#) (required if original transaction was not processed by Vantiv), [echeckSale](#), [echeckVerification](#), [forceCapture](#), [fraudCheck](#), [load](#), [sale](#), [unload](#)

This element is also a required child of the following Instruction-Based Dynamic Payout transaction types: [payFacCredit](#), [payFacDebit](#), [physicalCheckCredit](#), [physicalCheckDebit](#), [reserveCredit](#), [reserveDebit](#), [submerchantCredit](#), [submerchantDebit](#), [vendorCredit](#), [vendorDebit](#)

The amount element is an optional child of each of the following Parent Elements: [authReversal](#), [capture](#), [credit](#), [echeckCredit](#), [subscription](#)

NOTE: For all cases where the amount element is optional, except when a child of the subscription element, if you do not specify a value, the system uses the entire amount from the referenced (by littleTxnId) transaction. When amount is a child of the subscription element, if you do not specify a value, the amount defaults to the value defined in the referenced recurring plan (planCode element).

Attributes:

None

Child Elements:

None

4.28 androidpayResponse

The androidpayResponse element is an optional child of several transaction types and is returned in response messages, when the orderSource in the request is androidpay.

Parent Elements:

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

Attributes:

None

Child Elements (all optional):

[cryptogram](#), [expMonth](#), [expYear](#), [eciIndicator](#)

Example: androidpayResponse Structure

```
<androidpayResponse>
    <cryptogram>Payment Cryptogram</cryptogram>
    <expMonth>Expiration Month</expMonth>
    <expYear>Expiration Year</expYear>
    <eciIndicator>eCommerce Indicator</eciIndicator>
</androidpayResponse>
```

4.29 applepay

The `applepay` element is an optional child of several transaction types and takes the place of the `card` element in Apple Pay transaction where the merchant does not decrypt the (Apple Pay) `PKPaymentToken` prior to submitting the transaction. It contains child elements for the component parts of the `PKPaymentToken`.

Parent Elements:

[authorization](#), [registerTokenRequest](#), [sale](#)

Attributes:

None

Child Elements (all required):

[data](#), [header](#), [signature](#), [version](#)

Example: applepay Structure

```
<applepay>
  <data>User Name</data>
  <header>
    <applicationData>Base64 Hash of App Data Property</applicationData>
    <ephemeralPublicKey>Base64 Encoded Ephemeral Public Key</ephemeralPublicKey>
    <publicKeyHash>Base64 Hash of Public Merchant Key Cert</publicKeyHash>
    <transactionId>Hex Transaction Id</transactionId>
  </header>
  <signature>Signature of Payment and Header Data</signature>
  <version>Payment Token Version Info</version>
</applepay>
```

4.30 applepayResponse

The `applepayResponse` element is an optional child of several transaction types and is returned in response messages, when the request includes the `applepay` element.

Parent Elements:

`authorizationResponse`, `registerTokenResponse`, `saleResponse`

Attributes:

None

Child Elements:

`applicationPrimaryAccountNumber`, `applicationExpirationDate`, `currencyCode`,
`transactionAmount`, `cardholderName`, `deviceManufacturerIdentifier`, `paymentDataType`,
`onlinePaymentCryptogram`, `eciIndicator`

Example: `applepayResponse` Structure

```
<applepayResponse>
    <applicationPrimaryAccountNumber>App PAN</applicationPrimaryAccountNumber>
    <applicationExpirationDate>App PAN Exp Date</applicationExpirationDate>
    <currencyCode>Currency Code</currencyCode>
    <transactionAmount>Amount of Transaction</transactionAmount>
    <cardholderName>Name of cardholder</cardholderName>
    <deviceManufacturerIdentifier>Id of Device Mfr</deviceManufacturerIdentifier>
    <paymentDataType>Type of Payment Data</paymentDataType>
    <onlinePaymentCryptogram>Payment Cryptogram</onlinePaymentCryptogram>
    <eciIndicator>eCommerce Indicator</eciIndicator>
</applepayResponse>
```

IMPORTANT: The system never returns the `applicationPrimaryAccountNumber` element in a `registerTokenResponse` message, since this transaction type includes a token replacing the application PAN.

4.31 applicationData

The `applicationData` element is an optional child of the `header` element and provides the SHA-256 hash, hex encoded string of the original `PKPaymentRequest` of the Apple Pay transaction.

Type = Hex Encoded String; **minLength** = N/A; **maxLength** = 10000

Parent Elements:

[header](#)

Attributes:

None

Child Elements:

None

4.32 applicationExpirationDate

The `applicationExpirationDate` element is an optional child of the `applepayResponse` element and defines expiration date of the application primary account number.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[applepayResponse](#)

Attributes:

None

Child Elements:

None

4.33 applicationPrimaryAccountNumber

The applicationPrimaryAccountNumber element is an optional child of the applepayResponse element and defines the primary account number associated with the application.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

applepayResponse

Attributes:

None

Child Elements:

None

IMPORTANT: The system never returns the applicationPrimaryAccountNumber element (child of the applepayResponse element) in a registerTokenresponse message, since this transaction type includes a token replacing the application PAN.

4.34 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. It can also appear as the child of a deactivateResponse message where it specifies the value of the Gift Card prior to deactivation.

Type = Integer; totalDigits = 8

Parent Elements:

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

Attributes:

None

Child Elements:

None

4.35 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

4.36 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

4.37 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

4.38 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.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[cardholderAuthentication](#)

Attributes:

None

Child Elements:

None

4.39 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.

NOTE:

The authentication element and its child elements, user and password, are used to identify the merchant submitting transactions. They do not provide other access to platform or to the iQ reporting system.

Parent Elements:

[littleOnlineRequest](#), [littleRequest](#)

Attributes:

None

Child Elements:

Required: [user](#), [password](#)

Example: authentication Structure

```
<authentication>
    <user>User Name</user>
    <password>Password</password>
</authentication>
```

4.40 authenticationResult

The `authenticationResult` element is an optional child element of the `fraudResult` element. It defines the 3DS results from either Verified by Visa, MasterCard SecureCode, or Discover ProtectBuy. For a list of possible values, please refer to [3DS Authentication Result Codes](#) on page 835.

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

Parent Elements:

[fraudResult](#)

Attributes:

None

Child Elements:

None

4.41 authenticationTransactionId

The authenticationTransactionId element is an optional child of the cardholderAuthentication element. This element defines the Verified by Visa Transaction Id. This element is not used for Discover ProtectBuy transactions.

You must include this element for Visa transactions, when the orderSource element is set to **3dsAuthenticated**.

Type = Base 64 Encoded String; **minLength** = N/A; **maxLength** = 28

Parent Elements:

cardholderAuthentication

Attributes:

None

Child Elements:

None

4.42 authenticationValue

The authenticationValue element is an optional child 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).

You must include this element for Visa and MasterCard transactions, when the orderSource element is set to **3dsAuthenticated**, as well as MasterCard transactions if the orderSource element is set to **3dsAttempted**.

Type = Base 64 Encoded String; **minLength** = N/A; **maxLength** = 56

Parent Elements:

[cardholderAuthentication](#)

Attributes:

None

Child Elements:

None

4.43 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 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>
```

4.44 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
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: `orderId`, `amount`, `orderSource`, (choice of) `card`, `paypal`, `paypage`, `mpos`, `token`, or `applepay`, `cardholderAuthentication`

NOTE: The `cardholderAuthentication` child element is required only for 3-D Secure transactions.

Optional: `customerInfo`, `billToAddress`, `shipToAddress`, `billMeLaterRequest`, `processingInstructions`, `pos`, `customBilling`, `taxType`, `enhancedData`, `amexAggregatorData`, `allowPartialAuth`, `healthcareIIAS`, `merchantData`, `recyclingRequest`, `fraudFilterOverride`, `surchargeAmount`, `debtRepayment`, `recurringRequest`, `advancedFraudChecks`, `secondaryAmount`, `wallet`, `processingType`, `origOrderId`, `password`, `originalNetworkTransactionId`, `originalTransactionAmount`

4.45 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 littleOnlineResponse element or a batchResponse element.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	The response returns the same value submitted in the Authorization transaction. minLength = 1 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](#), [cardProductId](#) (see Note below), [authCode](#), [authorizationResponseSubCode](#) (see Note below), [approvedAmount](#), [accountInformation](#), [fraudResult](#), [billMeLaterResponseData](#), [tokenResponse](#), [enhancedAuthResponse](#), [accountUpdater](#), [recycling](#), [recurringResponse](#), [giftCardResponse](#), [applepayResponse](#), [cardSuffix](#), [androidpayResponse](#), [networkTransactionId](#)

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 Relationship Manager for additional information.

The **authorizationResponseSubCode** element is not used at this time.

4.46 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

4.47 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 76. Also, if you use the 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	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#)

Optional: [amount](#), [actionReason](#), [payPalNotes](#), [surchargeAmount](#)

NOTE: If you do not specify an amount child element, the system reverses the full amount from the associated Authorization transaction.

4.48 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	Yes	The response returns the same value submitted in the Authorization Reversal transaction. minLength = 1 maxLength = 25
customerId	String	No	The response returns the same value submitted in the Authorization Reversal transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the Authorization Reversal transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#)

NOTE: The postDate child element is returned only in responses to Online transactions.

4.49 availableBalance

The availableBalance element is a required child element of the fundingSource element and an optional child of the giftCardResponse element. It defines the outstanding available balance on the submitted prepaid or Gift Card. If the balance can not be determined, the element returns, “Not Available.”

Type = String; minLength = N/A; maxLength = 20

Parent Elements:

fundingSource, giftCardResponse

Attributes:

None

Child Elements:

None

NOTE: The fundingSource element and its child elements, type and availableBalance are associated with the Insights features (see Customer Insight Features on page 25.)

Please consult your Relationship Manager for additional information.

4.50 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 837.

Type = String; **minLength** = N/A; **maxLength** = 2

Parent Elements:

[fraudResult](#)

Attributes:

None

Child Elements:

None

4.51 balanceInquiry

The balanceInquiry element is the parent element for the transaction type that queries the available balance of a Gift Card.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (all Required)

[orderId](#), [orderSource](#), [card](#)

4.52 balanceInquiryResponse

The `balanceInquiryResponse` element is the parent element for information returned to you in response to a `balanceInquiry` 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
<code>id</code>	String	Yes	The response returns the same value submitted in the Balance Inquiry transaction. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	The response returns the same value submitted in the Balance Inquiry transaction. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	The response returns the same value submitted in the Balance Inquiry transaction. minLength = 1 maxLength = 25

Child Elements:

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

Optional: [postDate](#), [fraudResult](#), [giftCardResponse](#)

4.53 batchRequest

This is the root element for all LittleXML Batch requests.

Parent Elements:

[titleRequest](#)

Attributes:

NOTE: **Include the count and amount attributes for all transaction types included in the Batch. For example if you submit one Sale transaction of \$10, include numSales="1" and saleAmount="1000". You must always include an amount, if you include a count. Do not include corresponding attributes, if the Batch does not include the transaction type.**

Attribute Name	Type	Required?	Description
id	String	No	A unique string to identify this batchRequest within the system. minLength = N/A maxLength = 50
numAuths	Integer	No	Defines the total count of Authorization transactions in the batchRequest. minLength = N/A maxLength = N/A
authAmount	Integer	No	Defines the total dollar amount of Authorization transactions in the batchRequest. 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 batchRequest. minLength = N/A maxLength = N/A
authReversalAmount	Integer	No	Defines the total dollar amount of AuthReversal transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numGiftCardAuthReversals	Integer	No	Defines the total count of giftCardAuthReversal transactions in the batchRequest. minLength = N/A maxLength = N/A

Attribute Name	Type	Required?	Description
giftCardAuthReversalOriginalAmount	Integer	No	Defines the total dollar amount of giftCardAuthReversal transactions in the batchRequest. 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 batchRequest. minLength = N/A maxLength = N/A
captureAmount	Integer	No	Defines the total dollar amount of Capture transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numGiftCardCaptures	Integer	No	Defines the total count of giftCardCapture transactions in the batchRequest. minLength = N/A maxLength = N/A
giftCardCaptureAmount	Integer	No	Defines the total dollar amount of giftCardCapture transactions in the batchRequest. 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 batchRequest. minLength = N/A maxLength = N/A
creditAmount	Integer	No	Defines the total dollar amount of Credit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numGiftCardCredits	Integer	No	Defines the total count of giftCardCredit transactions in the batchRequest. minLength = N/A maxLength = N/A
giftCardCreditAmount	Integer	No	Defines the total dollar amount of giftCardCredit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10

Attribute Name	Type	Required?	Description
numForceCaptures	Integer	No	Defines the total count of Force Capture transactions in the batchRequest. minLength = N/A maxLength = N/A
forceCaptureAmount	Integer	No	Defines the total dollar amount of Force Capture transactions in the batchRequest. 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 batchRequest. minLength = N/A maxLength = N/A
saleAmount	Integer	No	Defines the total dollar amount of Sale transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numCaptureGivenAuths	Integer	No	Defines the total count of Capture Given Auth transactions in the batchRequest. minLength = N/A maxLength = N/A
captureGivenAuthAmount	Integer	No	Defines the total dollar amount of Capture Given Auth transactions in the batchRequest. 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 batchRequest. minLength = N/A maxLength = N/A
echeckSalesAmount	Integer	No	Defines the total dollar amount of eCheck Sale transactions in the batchRequest. 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 batchRequest. minLength = N/A maxLength = N/A

Attribute Name	Type	Required?	Description
echeckCreditAmount	Integer	No	Defines the total dollar amount of eCheck Credit transactions in the batchRequest. 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 batchRequest. minLength = N/A maxLength = N/A
echeckVerificationAmount	Integer	No	Defines the total dollar amount of eCheck Verification transactions in the batchRequest. 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 batchRequest. minLength = N/A maxLength = N/A
numEcheckPreNoteSale	Integer	No	Defines the total count of eCheck Prenotification Sale transactions in the batchRequest. minLength = N/A maxLength = N/A
numEcheckPreNoteCredit	Integer	No	Defines the total count of eCheck Prenotification Credit transactions in the batchRequest. minLength = N/A maxLength = N/A
numAccountUpdates	Integer	No	Defines the total count of Account Update transactions in the batchRequest. minLength = N/A maxLength = N/A
numTokenRegistrations	Integer	No	Defines the total count of Token Registration transactions in the batchRequest. minLength = N/A maxLength = N/A
numUpdateCardValidation	Integer	No	Defines the total count of Update Card Validation Number request transactions in the batchRequest. minLength = N/A maxLength = N/A
numCancelSubscriptions	Integer	No	Defines the total count of Cancel Subscription transactions in the batchRequest. minLength = N/A maxLength = N/A
numUpdateSubscriptions	Integer	No	Defines the total count of Update Subscription transactions in the batchRequest. minLength = N/A maxLength = N/A

Attribute Name	Type	Required?	Description
numCreatePlans	Integer	No	Defines the total count of Create Plan transactions in the batchRequest. minLength = N/A maxLength = N/A
numUpdatePlans	Integer	No	Defines the total count of Update Plan transactions in the batchRequest. minLength = N/A maxLength = N/A
numActivates	Integer	No	Defines the total count of (Gift Card) Activate transactions in the batchRequest. minLength = N/A maxLength = N/A
numDeactivates	Integer	No	Defines the total count of (Gift Card) Deactivate transactions in the batchRequest. minLength = N/A maxLength = N/A
activateAmount	Integer	No	Defines the total dollar amount of (Gift Card) Activate transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numLoads	Integer	No	Defines the total count of (Gift Card) Load transactions in the batchRequest. minLength = N/A maxLength = N/A
loadAmount	Integer	No	Defines the total dollar amount of (Gift Card) Load transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numUnloads	Integer	No	Defines the total count of (Gift Card) Unload transactions in the batchRequest. minLength = N/A maxLength = N/A
unloadAmount	Integer	No	Defines the total dollar amount of (Gift Card) Unload transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
numBalanceInquirys	Integer	No	Defines the total count of (Gift Card) Balance Inquiry transactions in the batchRequest. minLength = N/A maxLength = N/A

Attribute Name	Type	Required?	Description
numPayFacCredit	Integer	No	Defines the total count of PayFac Credit transactions in the batchRequest. minLength = N/A maxLength = N/A
numPayFacDebit	Integer	No	Defines the total count of PayFac Debit transactions in the batchRequest. minLength = N/A maxLength = N/A
numSubmerchantCredit	Integer	No	Defines the total count of Submerchant Credit transactions in the batchRequest. minLength = N/A maxLength = N/A
numSubmerchantDebit	Integer	No	Defines the total count of Submerchant Debit transactions in the batchRequest. minLength = N/A maxLength = N/A
numReserveCredit	Integer	No	Defines the total count of Reserve Credit transactions in the batchRequest. minLength = N/A maxLength = N/A
numReserveDebit	Integer	No	Defines the total count of Reserve Debit transactions in the batchRequest. minLength = N/A maxLength = N/A
numVendorCredit	Integer	No	Defines the total count of Vendor Credit transactions in the batchRequest. minLength = N/A maxLength = N/A
numVendorDebit	Integer	No	Defines the total count of Vendor Debit transactions in the batchRequest. minLength = N/A maxLength = N/A
numPhysicalCheckCredit	Integer	No	Defines the total count of Physical Check Credit transactions in the batchRequest. minLength = N/A maxLength = N/A
numPhysicalCheckDebit	Integer	No	Defines the total count of Physical Check Debit transactions in the batchRequest. minLength = N/A maxLength = N/A
numFundingInstructionVoid	Integer	No	Defines the total count of Funding Instruction Void transactions in the batchRequest. minLength = N/A maxLength = N/A

Attribute Name	Type	Required?	Description
payFacCreditAmount	Integer	No	Defines the total dollar amount of PayFac Credit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
payFacDebitAmount	Integer	No	Defines the total dollar amount of PayFac Debit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
submerchantCreditAmount	Integer	No	Defines the total dollar amount of Sub-merchant Credit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
submerchantDebitAmount	Integer	No	Defines the total dollar amount of Sub-merchant Debit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
reserveCreditAmount	Integer	No	Defines the total dollar amount of Reserve Credit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
reserveDebitAmount	Integer	No	Defines the total dollar amount of Reserve Debit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
vendorCreditAmount	Integer	No	Defines the total dollar amount of Vendor Credit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
vendorDebitAmount	Integer	No	Defines the total dollar amount of Vendor Debit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10

Attribute Name	Type	Required?	Description
physicalCheckCreditAmount	Integer	No	Defines the total dollar amount of Physical Check Credit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
physicalCheckDebitAmount	Integer	No	Defines the total dollar amount of Physical Check Debit transactions in the batchRequest. The decimal point is implied. For example, you enter \$25.00 as 2500. totalDigits = 10
merchantId	String	Yes	A unique string to identify the merchant within the system. minLength = N/A maxLength = 50 Note: International currencies are supported on a per merchantId basis.

Child Elements:

At least one of the following required: [activate](#), [authorization](#), [authReversal](#), [balanceInquiry](#), [cancelSubscription](#), [capture](#), [captureGivenAuth](#), [createPlan](#), [credit](#), [deactivate](#), [echeckCredit](#), [echeckPreNoteSale](#), [echeckRedeposit](#), [echeckSale](#), [echeckVerification](#), [forceCapture](#), [giftCardAuthReversal](#), [giftCardCapture](#), [giftCardCredit](#), [load](#), [registerTokenRequest](#), [sale](#), [unload](#), [updateCardValidationNumOnToken](#), [updatePlan](#), [updateSubscription](#), [payFacCredit](#), [payFacDebit](#), [reserveCredit](#), [reserveDebit](#), [submerchantCredit](#), [submerchantDebit](#), [vendorCredit](#), [vendorDebit](#)

4.54 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 Batch Request. minLength = N/A maxLength = 25
littleBatchId	Long	Yes	A unique value assigned by Vantiv to identify the batch. minLength = N/A maxLength = 19
merchantId	String	Yes	The response returns the same value submitted in the Batch Request. minLength = 1 maxLength = 50

Child Elements:

Required: [activateResponse](#), [authorizationResponse](#), [authReversalResponse](#), [captureResponse](#), [captureGivenAuthResponse](#), [createPlanResponse](#), [creditResponse](#), [deactivateResponse](#), [echeckCreditResponse](#), [echeckPreNoteCreditResponse](#), [echeckPreNoteSaleResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [forceCaptureResponse](#), [giftCardAuthReversalResponse](#), [giftCardCaptureResponse](#), [giftCardCreditResponse](#), [loadResponse](#), [registerTokenResponse](#), [saleResponse](#), [unloadResponse](#), [updateCardValidationNumOnTokenResponse](#), [cancelSubscriptionResponse](#), [updatePlanResponse](#), [updateSubscriptionResponse](#), [payFacCreditResponse](#), [payFacDebitResponse](#), [physicalCheckCreditResponse](#), [physicalCheckDebitResponse](#), [reserveCreditResponse](#), [reserveDebitResponse](#), [submerchantCreditResponse](#), [submerchantDebitResponse](#), [vendorCreditResponse](#), [vendorDebitResponse](#)

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.

4.55 beginningBalance

The beginningBalance element is an optional child of the giftCardResponse element. It defines the available balance on the submitted Gift Card prior to the requested operation.

NOTE: Although included in the schema, the beginningBalance, endingBalance, and cashBackAmount elements are not currently supported.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[giftCardResponse](#)

Attributes:

None

Child Elements:

None

4.56 billingDate

The `billingDate` element is an optional child of the `updateSubscription` element that defines the new date for the recurring billing when the scheduled date need to be changed.

Type = Date; **Format** = YYYY-MM-DD

Parent Elements:

[updateSubscription](#)

Attributes:

None

Child Elements:

None

4.57 billMeLaterRequest

The `billMeLaterRequest` element is the parent of several child elements used to define merchant, product type, terms and conditions, and other items for PayPal Credit authorization transactions, or where you must reference an external (to Vantiv) BML transaction.

NOTE:

As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

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>
```

4.58 billMeLaterResponseData

The `billMeLaterResponseData` element is the parent of several child elements used in the response XML for PayPal Credit authorization transactions.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

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>
```

4.59 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 Vantiv), [echeckPreNoteCredit](#), [echeckPreNoteSale](#), [echeckSale](#), [echeckVerification](#), [forceCapture](#), [fraudCheck](#), [sale](#), [updateSubscription](#)

Attributes:

None

Child Elements: (all optional)

[name](#), [firstName](#), [middleInitial](#), [lastName](#), [companyName](#), [addressLine1](#), [addressLine2](#), [addressLine3](#), [city](#), [state](#), [zip](#), [country](#), [email](#), [phone](#)

NOTE: You must submit the `name` element for `echeckSale` and `echeckCredit` transactions. If you do not submit the customer name in these eCheck transactions, we return the response 709 - Invalid Data.

You must submit the `name`, `country`, and `email` elements in `sale` transactions using the `sepaDirectDebit` method of payment. Failure to include the these elements results in a 901 - Missing Name Response Code, 903 - Missing Billing Country Code, or 905 - Missing Email Address respectively.

For an `eCheckVerification` transaction, you must submit the `firstName` and `lastName` elements instead of the `name` element (`middleInitial` is optional). For a corporate account you must include the `companyName` element in addition to the `firstName` and `lastName` elements. In both cases, you also must include the `address`, `city`, `state`, `zip` and `phone` 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 Middle Initial</middleInitial>
```

```
<lastName>Customer's Last 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>
```

4.60 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 in a registerTokenResponse.

For Discover, when included with Account Updater information in a payment transaction response, the bin element will have a value of **discov**.

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

4.61 bmlMerchantId

The `bmlMerchantId` element is a value assigned by PayPal Credit to identify the merchant within the PayPal Credit system.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

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

4.62 bmlProductType

The bmlProductType element is a value assigned by PayPal Credit to identify the merchant account type within the PayPal Credit system.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = String; **minLength** = 2; **maxLength** = 2

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

4.63 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

4.64 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.

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

Parent Elements:

[merchantData](#)

Attributes:

None

Child Elements:

None

4.65 cancelSubscription

The `cancelSubscription` element is the parent element for the transaction that cancels a subscription. You can use this element in either Online or Batch transactions.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

None

Child Elements:

Required: [subscriptionId](#)

4.66 cancelSubscriptionResponse

The cancelSubscriptionresponse element is the parent element for the response to a cancelSubscription transaction.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

None

Child Elements:

Required: [subscriptionId](#), [littleTxnId](#), [response](#), [message](#), [responseTime](#)

4.67 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

NOTE:

For CAT (Cardholder Activated Terminal) transactions, the **capability** element must be set to **magstripe**, the **cardholderId** element must be set to **nopin**, and the **catLevel** element must be set to **self service**.

4.68 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	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 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 Note: If you settle in a currency other than US or Canada, you cannot use partial captures.

Child Elements:

Required: [littleTxnId](#), [paypalOrderComplete](#) (required only if closing a PayPal order)

Optional: [amount](#), [enhancedData](#), [payPalNotes](#), [pin](#), [processingInstructions](#), [secondaryAmount](#), [surchargeAmount](#)

NOTE: If you do not specify an amount, the system uses the full amount from the associated Authorization transaction.

4.69 captureAmount

The `captureAmount` element is a required child of the `giftCardCapture` element and defines the amount of the capture (deposit). This value must be less than or equal to the value of the `originalAmount` element. Setting this element to a value less than the `originalAmount` implies that this is a partial capture.

Type = Integer; totalDigits = 12

Parent Elements:

`giftCardCapture`

Attributes:

None

Child Elements:

None

4.70 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 our system. You can use this element in either Online or Batch transactions.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: `orderId`, `authInformation`, `amount`, `orderSource`, choice of `card`, `token`, `mpos`, `paypage`, or `applepay`

Optional: `billToAddress`, `shipFromPostalCode`, `customBilling`, `taxType`, `enhancedData`, `processingInstructions`, `pos`, `amexAggregatorData`, `merchantData`, `secondaryAmount`, `surchargeAmount`, `debtRepayment`, `processingType`, `origOrderId`, `password`, `originalNetworkTransactionId`, `originalTransactionAmount`

NOTE: If you do not specify an `amount` child element, the system uses the full amount from the associated Authorization transaction.

4.71 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
<code>id</code>	String	Yes	The response returns the same value submitted in the Capture Given Auth transaction. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	The response returns the same value submitted in the Capture Given Auth transaction. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	The response returns the same value submitted in the Capture Given Auth transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [tokenResponse](#), [giftCardResponse](#), [applepayResponse](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.72 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
<code>id</code>	String	Yes	The response returns the same value submitted in the capture transaction. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	The response returns the same value submitted in the capture transaction. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	The response returns the same value submitted in the capture transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [accountUpdater](#), [giftCardResponse](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.73 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:

[activate](#), [activateReversal](#), [accountUpdate](#), [authorization](#), [balanceInquiry](#), [captureGivenAuth](#), [credit](#), [deactivate](#), [deactivateReversal](#), [depositReversal](#), [forceCapture](#), [giftCardAuthReversal](#), [giftCardCapture](#), [giftCardCredit](#), [load](#), [loadReversal](#), [refundReversal](#), [sale](#), [unload](#), [unloadReversal](#), [updateSubscription](#)

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](#)

For Gift Card transactions (Optional): [pin](#)

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>
```

Example: card Structure - Gift Card (Card-Not-Present)

```
<card>
```

```
<type>GC</type>
<number>Account Number</number>
<expDate>Expiration Date</expDate>
<cardValidationNum>Card Validation Number</cardValidationNum>
<pin>Pin Number</pin>
</card>
```

4.74 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

4.75 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`, `authenticationTransactionId`, and `authenticatedByMerchant` elements). The remaining child element, `customerIpAddress`, as well as the `authenticationTransactionId`, are used in PayPal Credit transactions. The `customerIpAddress` element can also be used to supply the customer IP Address by merchants enabled for American Express Advanced AVS services.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Parent Elements:

`authorization`, `sale`

Attributes:

None

Child Elements:

`authenticationValue`, `authenticationTransactionId`, `customerIpAddress`, `authenticatedByMerchant`

Example: cardholderAuthentication Structure

```
<cardholderAuthentication>
    <authenticationValue>BwABBHQ1AgAAAAAgJDUCAAAAAA=</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.

4.76 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

NOTE: For CAT (Cardholder Activated Terminal) transactions, the **capability** element must be set to **magstripe**, the **cardholderId** element must be set to **nopin**, and the **catLevel** element must be set to **self service**.

4.77 cardholderName

The `cardholderName` element is an optional child of the `applepayResponse` element and provides the name of the cardholder whose card initiated the Apple Pay transaction.

Type = String; **minLength** = N/A; **maxLength** = 512

Parent Elements:

[applepayResponse](#)

Child Elements:

None

4.78 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](#)

4.79 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.

4.80 cardSuffix

The `cardSuffix` element is an optional child of both the `authorizationResponse` and `saleResponse` elements. It provides the last four digits of the actual PAN for Apple Pay transactions, when the underlying card is either Visa or MasterCard.

Type = String; **minLength** = 3; **maxLength** = 6

Parent Elements:

`authorizationResponse`, `saleResponse`

Attributes:

None

Child Elements:

None

4.81 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.

When you submit the CVV2/CVC2/CID in a `registerTokenRequest`, the platform encrypts and stores the value on a temporary basis for later use in a tokenized Auth/Sale transaction submitted without the value. This is done to accommodate merchant systems/workflows where the security code is available at the time of token registration, but not at the time of the Auth/Sale. If for some reason you need to change the value of the security code supplied at the time of the token registration, use an `updateCardValidationNumOnToken` transaction. To use the stored value when submitting an Auth/Sale transaction, set the `cardValidationNum` value to 000.

The `cardValidationNum` element is an optional child of the `virtualGiftCardResponse` element, where it defines the value of the validation Number associated with the Virtual Gift Card requested.

NOTE:

The use of the `cardValidationNum` element in the `registertokenRequest` only applies when you submit an `accountNumber` element.

Type = String; **minLength** = N/A; **maxLength** = 4

Parent Elements:

[card](#), [paypage](#), [token](#), [registerTokenRequest](#), [updateCardValidationNumOnToken](#), [virtualGiftCardResponse](#)

Attributes:

None

Child Elements:

None

4.82 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 841.

Type = String; **minLength** = N/A; **maxLength** = 2

Parent Elements:

[fraudResult](#)

Attributes:

None

Child Elements:

None

4.83 cashBackAmount

The cashBackAmount element is an optional child of the giftCardResponse element. It defines the Cash Back amount provided to the Gift Card holder.

NOTE: Although included in the schema, the beginningBalance, endingBalance, and cashBackAmount elements are not currently supported.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[giftCardResponse](#)

Attributes:

None

Child Elements:

None

4.84 catLevel

The **catLevel** element is an optional child of the **pos** element, which describes the capability of the Cardholder Activated Terminal. Although optional in the schema, it is required for all CAT transactions.

Type = String (Enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[pos](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
self service	Cardholder Activated Terminal level

4.85 ccdPaymentInformation

The `ccdPaymentInformation` element is an optional child of the `echeck` element. This element is intended for use by PayFacs using Instruction Based Dynamic Payout to submit a description of the transaction. The description will appear in the extended detail section of the receiver's bank statement, if that section is supported by the receiver's bank.

Type = String; **minLength** = N/A; **maxLength** = 80

Parent Elements:

[accountInfo](#), [echeck](#)

Attributes:

None

Child Elements:

None

4.86 chargeback

The chargeback element is an optional child of the filtering element. To disable the chargeback filtering operation for a selected transaction include the chargeback element with a setting of **false**.

Type = Boolean; **Valid Value** = false

NOTE: Although included in the schema, the <chargeback> element is not supported. To override the chargeback filter for a selected transaction, use the fraudFilterOverride flag (see [fraudFilterOverride](#) on page 538). Please consult your Relationship Manager for additional information.

Parent Elements:

[filtering](#)

Attributes:

None

Child Elements:

None

4.87 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

4.88 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

4.89 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

4.90 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

4.91 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.niggp.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”.

4.92 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

4.93 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.

All Country Codes are 2-character except for the United States, which accepts both US and USA.

The country element is required for sale transactions using the sepaDirectDebit method of payment. Failure to include the name element results in a 903 - Missing Billing Country Code Response Code.

Parent Elements:

[billToAddress](#), [shipFromPostalCode](#)

Attributes:

None

Child Elements:

None

4.94 createAddOn

The `createAddOn` element is the parent of several child elements used to define an additional charge added to a new or existing subscription.

Parent Elements:

[subscription](#), [updateSubscription](#)

Attributes:

None

Child Elements (all Required):

[addOnCode](#), [name](#), [amount](#), [startDate](#), [endDate](#)

Example: `customerInfo` Structure

```
<createAddOn>
    <addOnCode>Add On Reference Code</addOnCode>
    <name>Name of Add On</name>
    <amount>Amount of Add On</amount>
    <startDate>Start Date of Add On Charge</startDate>
    <endDate>End Date of Add On Charge</endDate>
</createAddOn>
```

4.95 createDiscount

The `createDiscount` element is the parent of several child elements used to define a discount to be applied to a new or existing subscription.

Parent Elements:

[subscription](#), [updateSubscription](#)

Attributes:

None

Child Elements (all Required):

`discountCode`, [name](#), [amount](#), [startDate](#), [endDate](#)

Example: `customerInfo` Structure

```
<createDiscount>
    <discountCode>Discount Reference Code</discountCode>
    <name>Name of Discount</name>
    <amount>Amount of Discount</amount>
    <startDate>Start Date of Discount</startDate>
    <endDate>End Date of Discount</endDate>
</createDiscount>
```

4.96 createPlan

The `createPlan` element is the parent of several child element that define the attributes of a recurring payment plan. You associate Plans with subscriptions to define the billing behavior for the recurring payment.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

None

Child Elements:

[planCode](#), [name](#), [description](#), [intervalType](#), [amount](#), [numberOfPayments](#), [trialNumberOfIntervals](#), [trialIntervalType](#), [active](#)

Example: createPlan Structure

```
<createPlan>
    <planCode>Plan Reference Code</planCode>
    <name>Name of Plan</name>
    <description>Description of Plan</description>
    <intervalType>The Type of Interval</intervalType>
    <amount>Amount of Recurring Payment</amount>
    <numberOfPayments>1 to 99</numberOfPayments>
    <trialNumberOfIntervals>Number of Trial Period
    Intervals</trialNumberOfIntervals>
    <trialIntervalType>Type of Trial Period Interval</trialIntervalType>
    <active>true or false</active>
</createPlan>
```

4.97 createPlanResponse

The `createPlanResponse` element is the parent element for the response to a `createPlan` transaction.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

None

Child Elements:

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

4.98 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:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

For credits to transactions processed by Vantiv (Required): [littleTxnId](#)

For credits to transactions processed by Vantiv (Optional): [amount](#), [payPalNotes](#), [pin](#), [secondaryAmount](#), [surchargeAmount](#)

NOTE: If you do not specify an `amount`, the system uses the full amount from the associated settlement transaction.

You must include the `amount` element for credits to transactions not processed by Vantiv

For credits to transactions not processed by Vantiv (Required): [orderId](#), [amount](#), [orderSource](#), choice of [card](#), [token](#), [paypage](#), [mpos](#), [paypal](#), or [applepay](#)

For credits to transactions not processed by Vantiv (Optional): [billToAddress](#), [billMeLaterRequest](#), [amexAggregatorData](#)

For both transaction types (Optional): [customBilling](#), [taxType](#), [enhancedData](#), [processingInstructions](#), [merchantData](#), [actionReason](#), [pos](#)

4.99 creditAmount

The creditAmount element is a required child of the giftCardCredit transaction, where it specifies the amount of the refund. Supply the value in cents without a decimal point. For example, a value of 1995 signifies \$19.95.

Type = Integer; **totalDigits** = 12

Parent Elements:

[giftCardCredit](#)

Attributes:

None

Child Elements:

None

4.100 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

4.101 creditLittleTxnId

The `creditLittleTxnId` element is the Transaction Id (`littleTxnId`) of a Credit transaction automatically submitted under the following conditions:

- You submitted a Void transaction to halt the recycling of a declined Sale transaction by the Recovery/Recycling Engine.
- The Sale transaction has already been approved and captured.
- Your Recovery/Recycling Engine configuration enables automatic refunds.
- The system has successfully submitted a Credit transaction on your behalf.

Type = Long; **minLength** = N/A; **maxLength** = 19

Parent Elements:

[recycling](#)

Attributes:

None

Child Elements:

None

4.102 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
<code>id</code>	String	Yes	The response returns the same value submitted in the credit transaction. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	The response returns the same value submitted in the credit transaction. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	The response returns the same value submitted in the credit transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [tokenResponse](#), [giftCardResponse](#), [applepayResponse](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.103 cryptogram

The `cryptogram` element is an optional child of the `androidResponse` element and provides the BASE64 Encoded signature cryptogram associated with the Android Pay transaction.

Type = Base 64 Encoded String; **minLength** = N/A; **maxLength** = 56

Parent Elements:

[androidpayResponse](#)

Attributes:

None

Child Elements:

None

4.104 currencyCode

The currencyCode element is an optional child of the `applepayResponse` element and provides the 3-character code for the currency used in the transaction.

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

Parent Elements:

[applepayResponse](#)

Attributes:

None

Child Elements:

None

4.105 customAttribute1

The `customAttribute1` through `customAttribute5` elements are optional children of the `advancedFraudChecks` element. These elements allow users of Advanced Fraud Tools with self-serve rules to submit additional custom data to ThreatMetrix for inclusion in the fraud evaluation process. For example, if you assigned a certain attribute to your customers for segmentation purposes, you might wish to submit the assigned value and also establish a ThreatMetrix rule so the value was included in the evaluation.

NOTE:

Once you decide to use a particular custom attribute for a particular value set and establish a rule for its evaluation, you

Type = String; **minLength** = 1; **maxLength** = 200

Parent Elements:

[advancedFraudChecks](#)

Attributes:

None

Child Elements:

None

4.106 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 system uses the default.

NOTE: If you submit a `captureGivenAuth` transaction with a `customBilling` element and a matching Authorization is found (see [Capture Given Auth Transaction](#) on page 78), the system uses the `customBilling` information from the Authorization and discards the information from the `captureGivenAuth`.

Parent Elements:

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

Attributes:

None

Child Elements:

Required for card-not-present transactions: [phone](#), or [url](#)

NOTE: Please consult your 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>
```

4.107 customIdentifier

The `customIdentifier` element is an optional child of Sub-merchant funding instruction request transaction types, as well as eCheck Sale/Credit/Redeposit requests. PayFacs/Merchants can use this element to specify a Billing Descriptor to appear on the bank statements of the parties involved in the funds transfer or eCheck transaction.

NOTE: The information you provide in this element populates the Individual ID field of the ACH Record. The use of this field and its appearance on bank statements is at the discretion of the bank producing the statement.

Type = String; **minLength** = N/A; **maxLength** = 15

Parent Elements:

[echeckCredit](#), [echeckRedeposit](#), [echeckSale](#), [submerchantCredit](#), [submerchantDebit](#)

Attributes:

None

Child Elements:

None

4.108 customerInfo

The `customerInfo` element is the parent of several child elements used to define customer information for PayPal Credit transactions.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

[ssn](#), [dob](#), [customerRegistrationDate](#), [customerType](#), [incomeAmount](#), [employerName](#), [customerWorkTelephone](#), [residenceStatus](#), [yearsAtResidence](#), [yearsAtEmployer](#)

NOTE: Under certain conditions PayPal Credit requires you to include `ssn`, `dob`, `customerType` and `customerRegistrationDate`.

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>
```

4.109 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 PayPal Credit transactions or to supply the customer IP Address by merchants enabled for American Express Advanced AVS services.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

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

4.110 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

4.111 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 PayPal Credit transactions.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

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

4.112 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 PayPal Credit transactions.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = Choice (Enum); **Enumerations** = New or Existing

NOTE: In order for a PayPal Credit transaction to succeed, you must include this element if the customer does not have a PayPal Credit account.

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.113 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 PayPal Credit transactions.

NOTE:

As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.114 data

The data element is a required child of the applepay element. It is the payment data dictionary, BASE64 encoded string from the PKPaymentToken.

Type = String; **minLength** = N/A; **maxLength** = 2000

Parent Elements:

[applepay](#)

Attributes:

None

Child Elements:

None

4.115 deactivate

The deactivate element is the parent element for the transaction type that deactivate a Gift Card.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (all Required)

[orderId](#), [orderSource](#), [card](#)

4.116 deactivateResponse

The deactivateResponse element is the parent element for information returned to you in response to a **deactivate** 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	Yes	The response returns the same value submitted in the Deactivate transaction. minLength = 1 maxLength = 25
customerId	String	No	The response returns the same value submitted in the Deactivate transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the Deactivate transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [fraudResult](#), [giftCardResponse](#)

4.117 deactivateReversal

The deactivateReversal element is the parent element for the transaction type that reverses the deactivation of a Gift Card.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (Required)

[card](#), [originalRefCode](#), [originalAmount](#), [originalTxnTime](#), [originalSystemTraceId](#), [originalSequenceNumber](#)

Child Elements: (Optional)

[littleTxnId](#)

4.118 deactivateReversalResponse

The deactivateReversalResponse element is the parent element for information returned to you in response to an deactivateReversal transaction.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [giftCardResponse](#)

4.119 debtRepayment

The debtRepayment element is an optional child of authorization, captureGivenAuth, forceCapture, and sale transactions. You use this flag only if the method of payment is Visa and the transaction is a for a debt repayment. Also, your MCC must be either 6012 or 6051. If you set this flag to true and you are not one of the allowed MCCs, the system declines the transaction with a Response Reason Code of 852 - Debt Repayment only allowed for VI transactions on MCCs 6012 and 6051.

Type = Boolean; **Valid Values** = true or false (default)

Parent Elements:

authorization, captureGivenAuth, forceCapture, sale

Attributes:

None

Child Elements:

None

4.120 deleteAddOn

The `deleteAddOn` element is the parent element used to define an Add On to be removed from an existing subscription.

Parent Elements:

[updateSubscription](#)

Attributes:

None

Child Elements (all Required):

[addOnCode](#)

Example: deleteAddOn Structure

```
<deleteAddOn>
    <addOnCode>Add On Reference Code</addOnCode>
</deleteAddOn>
```

4.121 deleteDiscount

The `deleteDiscount` element is the parent element used to define a discount to be removed from an existing subscription.

Parent Elements:

[updateSubscription](#)

Attributes:

None

Child Elements (all Required):

[discountCode](#)

Example: deleteDiscount Structure

```
<deleteDiscount>
    <discountCode>Discount Reference Code</discountCode>
</deleteDiscount>
```

4.122 deliveryType

The `deliveryType` element is an optional child of the `enhancedData` element and defines the shipping method used for delivery of the product.

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.

4.123 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

4.124 depositReversal

The `depositReversal` element is the parent element for a Gift Card specific transaction type that reverses a `giftCardCapture` or `sale` transaction.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (Required)

[card](#), [originalRefCode](#), [originalAmount](#), [originalTxnTime](#), [originalSystemTraceId](#), [originalSequenceNumber](#)

Child Elements: (Optional)

[littleTxnId](#)

4.125 depositReversalResponse

The `depositReversalResponse` element is the parent element for information returned to you in response to an `depositReversal` transaction.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#), [giftCardResponse](#)

Optional: [postDate](#)

4.126 description

The `description` element is an optional child of the `createPlan` element and provides a text description of the Plan.

Type = String; **minLength** = N/A; **maxLength** = 100

Parent Elements:

[createPlan](#)

Attributes:

None

Child Elements:

None

4.127 descriptor

The **descriptor** element is an optional 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

4.128 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 <countryTypeEnum> 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

4.129 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

4.130 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>
```

4.131 deviceManufacturerIdentifier

The `deviceManufacturerIdentifier` element is an optional child of the `applepayResponse` element and defines the manufacturer of the device originating the transaction.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[applepayResponse](#)

Attributes:

None

Child Elements:

None

4.132 deviceReputationScore

The `deviceReputationScore` element is an optional child of the `advancedFraudResults` element and specifies score resulting from the ThreatMetrix analysis of the customer device.

Type = Integer; **totalDigits** = 8

Parent Elements:

[advancedFraudResults](#)

Attributes:

None

Child Elements:

None

4.133 deviceReviewStatus

The `deviceReviewStatus` element is a required child of the `advancedFraudResults` element and specifies the results of the comparison of the `deviceReputationScore` against the threshold levels configured for the merchant. Typical values are: pass, fail, review, unavailable, and `invalid_session`.

NOTE: The system returns `invalid_session` if you submitted a session Id using an invalid prefix.

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

Parent Elements:

[advancedFraudResults](#)

Attributes:

None

Child Elements:

None

4.134 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

4.135 discountCode

The `discountCode` element is the identifier of a defined discount. You use this element when creating, updating, and deleting a discount applied to a subscription.

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

Parent Elements:

[createDiscount](#), [updateDiscount](#), [deleteDiscount](#)

Attributes:

None

Child Elements:

None

4.136 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 PayPal Credit transactions.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = Date; **Format** = YYYY-MM-DD

NOTE: In order for a PayPal Credit transaction to succeed, you must include this element if:

- the customer does not have a PayPal Credit account
or
- the customer has a PayPal Credit account, but the account has not been authenticated.

You do not need to include this element if the PayPal Credit account has been authenticated.

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.137 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

4.138 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 our system) elements. It contains child elements used to provide details concerning the eCheck account.

Parent Elements:

echeckCredit, echeckPreNoteCredit, echeckPreNoteSale, echeckSale, echeckVerification

Attributes:

None

Child Elements:

Required: [accType](#), [accNum](#), [routingNum](#)

Optional: [checkNum](#), [ccdPaymentInformation](#)

Example: echeck Structure

```
<echeck>
  <accType>Account Type Abbreviation</accType>
  <accNum>Account Number</accNum>
  <routingNum>Routing Number</routingNum>
  <checkNum>Check Number</checkNum>
  <ccdPaymentInformation>Payment Description</ccdPaymentInformation>
</echeck>
```

4.139 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

4.140 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](#), [littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

For credits to transactions processed by Vantiv (Required): [littleTxnId](#)

For credits to transactions processed by Vantiv (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 Vantiv (Required): [orderId](#), [amount](#), [orderSource](#), [billToAddress](#), [echeckOrEcheckToken](#) (allows the substitution of either the `echeck` or `echeckToken` elements)

For credits to transactions not processed by Vantiv (Optional): [merchantData](#)

For both (Optional): [customBilling](#), [secondaryAmount](#), [customIdentifier](#)

4.141 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	Yes	The response returns the same value submitted in the echeckCredit transaction. minLength = 1 maxLength = 25
customerId	String	No	The response returns the same value submitted in the echeckCredit transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the echeckCredit transaction. minLength = 1 maxLength = 25

Child Elements: (all Required)

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

Optional: [postDate](#), [accountUpdater](#)

NOTE: The **postDate** child element is returned only in responses to Online transactions.

4.142 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>
```

4.143 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](#)

4.144 echeckPreNoteCredit

The echeckPreNoteCredit element is the parent element for all eCheck Prenotification Credit transactions. You use this transaction type to perform an eCheck Prenotification, when the subsequent eCheck transaction will be an eCheck Credit transaction. You can use this element only in Batch transactions. This transaction type is only supported for US transactions.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [orderSource](#), [billToAddress](#), [echeck](#)

Optional: [merchantData](#)

4.145 echeckPreNoteCreditResponse

The echeckPreNoteCreditResponse element is the parent element for information returned to you in response to an echeckPreNoteCredit transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements: (all Required)

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

4.146 echeckPreNoteSale

The echeckPreNoteSale element is the parent element for all eCheck Prenotification Sale transactions. You use this transaction type to perform an eCheck Prenotification, when the subsequent eCheck transaction will be an eCheck Sale transaction. You can use this element only in Batch transactions. This transaction type is only supported for US transactions.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [orderSource](#), [billToAddress](#), [echeck](#)

Optional: [merchantData](#)

4.147 echeckPreNoteSaleResponse

The echeckPreNoteSaleResponse element is the parent element for information returned to you in response to an echeckPreNoteSale transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements: (all Required)

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

4.148 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](#), [littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#)

Optional: [echeckOrEcheckToken](#) (allows the substitution of either the echeck or echeckToken elements), [merchantData](#), [customIdentifier](#)

4.149 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	Yes	The response returns the same value submitted in the echeckSale transaction. minLength = 1 maxLength = 25
customerId	String	No	The response returns the same value submitted in the echeckSale transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the echeckSale 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.

4.150 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](#), [littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 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)

NOTE: The value for the **orderSource** element must be one of the following:
telephone, **ecommerce**, **recurringtel**, or **echeckppd**.

Optional: [shipToAddress](#), [verify](#), [customBilling](#), [merchantData](#), [secondaryAmount](#), [customIdentifier](#)

4.151 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	Yes	The response returns the same value submitted in the echeckSale transaction. minLength = 1 maxLength = 25
customerId	String	No	The response returns the same value submitted in the echeckSale transaction. minLength = N/A maxLength = 50
reportGroup	String	Yes	The response returns the same value submitted in the echeckSale transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [litleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [accountUpdater](#), [tokenResponse](#)

NOTE: The **postDate** child element is returned only in responses to Online transactions.

4.152 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: [littleToken](#), [routingNum](#), [accType](#)

Optional: [checkNum](#)

Example: echeck Structure

```
<echeckToken>
    <littleToken>Token</littleToken>
    <routingNum>Routing Number</routingNum>
    <accType>Account Type Abbreviation</accType>
    <checkNum>Check Number</checkNum>
</echeckToken>
```

4.153 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. This transaction type is only supported for US transactions.

Parent Elements:

[batchRequest](#), [littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 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: [merchantData](#)

4.154 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	Yes	The response returns the same value submitted in the capture transaction. minLength = 1 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](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#)

NOTE: The **postDate** child element is returned only in responses to Online transactions.

4.155 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:

[litleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [litleTxnId](#)

4.156 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	Yes	The response returns the same value submitted in the void transaction. minLength = 1 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

Child Elements: (all Required)

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

4.157 eciIndicator

The `eciIndicator` element is an optional child of the `applepayResponse` and `androidpayResponse` elements and specifies electronic commerce indicator associated with an Apple Pay/Android Pay transaction.

Type = String; **minLength** = N/A; **maxLength** = 2

Parent Elements:

`applepayResponse`, `androidpayResponse`

Attributes:

None

Child Elements:

None

4.158 email

The **email** element defines the email address of the customer in both the **billToAddress** and **shipToAddress** elements.

NOTE: The **email** element is required for sale transactions using the **sepaDirectDebit** method of payment. Failure to include the **name** element results in a 905 - Missing Email Address Response Code.

Type = String; **minLength** = N/A; **maxLength** = 100

Parent Elements:

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

Attributes:

None

Child Elements:

None

4.159 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 PayPal Credit transactions.

NOTE:

As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

4.160 encryptedTrack

The `encryptedTrack` element is a required child of the `mpos` element. This element provides the encrypted track data resulting from a card swipe using a ROAM device.

Type = String; **minLength** = 1; **maxLength** = 1028

Parent Elements:

[mpos](#)

Attributes:

None

Child Elements:

None

4.161 endDate

The endDate element is a optional child of both the createAddOn and createDiscount element, where it specifies either the ending date of the Add On charge or the ending date of the discount.

Type = Date; **Format** = YYYY-MM-DD

Parent Elements:

[createAddOn](#), [createDiscount](#)

Attributes:

None

Child Elements:

None

4.162 endingBalance

The endingBalance element is an optional child of the giftCardResponse element. It defines the available balance on the submitted Gift Card after the requested operation.

NOTE: Although included in the schema, the beginningBalance, endingBalance, and cashBackAmount elements are not currently supported.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[giftCardResponse](#)

Attributes:

None

Child Elements:

None

4.163 endpoint

The endpoint element is a required child of the networkResponse element. It defines the card network (i.e., Visa, MasterCard, Discover, or American Express) acting as an endpoint for the submitted transaction.

Type = String; **minLength** = N/A; **maxLength** = 256

Parent Elements:

[networkResponse](#)

Attributes:

None

Child Elements:

None

4.164 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. Depending upon the card used, other elements can indicate affluence, card product type, prepaid card type, reloadability and whether or not it is a virtual account number.

Parent Elements:

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

Attributes:

None

Child Elements: (all Optional)

[fundingSource](#), [affluence](#), [issuerCountry](#), [cardProductType](#), [virtualAccountNumber](#), [networkResponse](#)

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>
```

Example: enhancedAuthResponse - with virtualAccountNumber

```
<enhancedAuthResponse>
    <virtualAccountNumber>true or false</virtualAccountNumber>
</enhancedAuthResponse>
```

Example: enhancedAuthResponse - with networkResponse

```
<enhancedAuthResponse>
    <networkResponse>
        <endpoint>VISA</endpoint>
        <networkField fieldNumber="4" fieldName="Transaction Amount">
            <fieldValue>00000000962</fieldValue>
        </networkField>
        <networkField fieldNumber="44" fieldName="Additional Response Data">
            <networkSubField fieldNumber="0">
                <fieldValue></fieldValue>
            </networkSubField>
            <networkSubField fieldNumber="1">
                <fieldValue>5</fieldValue>
            </networkSubField>
        </networkField>
    </networkResponse>
</enhancedAuthResponse>
```

4.165 enhancedData

The enhancedData element allows you to specify extra information concerning a transaction in order to qualify for certain purchasing interchange rates. The following tables provide information about required elements you must submit to achieve Level 2 or Level 3 Interchange rates for Visa and MasterCard.

- For MasterCard:
 - The transaction must be taxable.
 - The tax charged must be between 0.1% and 30% of the transaction amount.
 - For Level 3, the transaction must use a corporate, business, or purchasing card.

NOTE: You can qualify for MasterCard Level 2 rates without submitting the total tax amount (submit 0) if your MCC is one of the following: 4111, 4131, 4215, 4784, 8211, 8220, 8398, 8661, 9211, 9222, 9311, 9399, 9402.

- You must include at least one line item with amount, description, and quantity defined.

TABLE 4-1 MasterCard Level 2/Level 3 Data Requirements

MasterCard Level 2 Data	MasterCard Level 3 Data	LittleXML Element (child of enhancedData unless noted)
Customer Code (if supplied by customer)	Customer Code (if supplied by customer)	customerReference
Card Acceptor Tax ID	Card Acceptor Tax ID	cardAcceptorTaxId (child of detailTax)
Total Tax Amount	Total Tax Amount	salesTax
	Product Code	productCode (child of lineItemData)
	Item Description	itemDescription (child of lineItemData)
	Item Quantity	quantity (child of lineItemData)
	Item Unit of Measure	unitOfMeasure (child of lineItemData)
	Extended Item Amount	lineItemTotal (child of lineItemData) or lineItemTotalWithTax (child of lineItemData)

In addition to the requirements below, please be aware of the following:

- For Visa:
 - The transaction must be taxable, except for Level 3 if tax exempt.
 - The tax charged must be between 0.1% and 22% of the transaction amount, except as noted.
 - For Level 3, the transaction must use a corporate or purchasing card.

TABLE 4-2 Visa Level 2/Level 3 Data Requirements

Visa Level 2 Data	Visa Level 3 Data	LittleXML Element (child of enhancedData unless noted)
Sales Tax	Sales Tax*	salesTax
	Discount Amount	discountAmount
	Freight/Shipping Amount	shippingAmount
	Duty Amount	dutyAmount
	Item Sequence Number	itemSequenceNumber (child of lineItemData)
	Item Commodity Code	commodityCode (child of lineItemData)
	Item Description	itemDescription (child of lineItemData)
	Product Code	productCode (child of lineItemData)
	Quantity	quantity (child of lineItemData)
	Unit of Measure	unitOfMeasure (child of lineItemData)
	unit Cost	unitCost (child of lineItemData)
	Discount per Line Item	itemDiscountAmount (child of lineItemData)
	Line Item Total	lineItemTotal (child of lineItemData)

* Sales Tax is no longer required for Level 3, if you are tax exempt. In this case omit the <salesTax> element entirely.

NOTE: We 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, Vantiv strongly recommends that you adhere to the guidelines detailed above.

For Discover transactions, the interchange rate is not impacted by the presence of Level 2 or Level 3 data.

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>
</enhancedData>
```

```
</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>
```

4.166 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.)
completread	complete magnetic stripe read and transmitted

4.167 ephemeralPublicKey

The `ephemeralPublicKey` element is a required child of the `header` element and provides the BASE64 Encoded string of the ephemeral public key bytes from the Apple Pay transaction.

Type = Base 64 Encoded String; **minLength** = N/A; **maxLength** = 400

Parent Elements:

[header](#)

Attributes:

None

Child Elements:

None

4.168 expDate

The `expDate` element is a child of the `card`, `token`, `paypage` elements, which specifies the expiration date of the card (format: mmyy) 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.**

We recommend all merchant with recurring and/or installment payments participate in the Account Updater program.

4.169 expMonth

The `expMonth` element is an optional child of the `androidpayResponse` element, which specifies the month of expiration of the network token (format: mm).

Type = String; **minLength** = 2; **maxLength** = 2

Parent Elements:

[androidpayResponse](#)

Attributes:

None

Child Elements:

None

4.170 expYear

The `expYear` element is an optional child of the `androidpayResponse` element, which specifies the year of expiration of the network token (format: yyyy).

Type = String; **minLength** = 4; **maxLength** = 4

Parent Elements:

[androidpayResponse](#)

Attributes:

None

Child Elements:

None

4.171 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.”

IMPORTANT: When using Account Updater (any variation), you must always code to receive the extendedCardResponse element and its children. We always return this information whenever applicable regardless of whether you receive other account updater information in the transaction response message.

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>
```

4.172 fieldValue

The `fieldValue` element is a child of both the `networkField` and `networkSubField` elements. The `fieldValue` element is required when a child of the `networkField` element and no subfield exists. This element is always required when used as a child of the `networkSubField` element. It provides the raw data for the designated field, extracted from the network ISO 8583 response message.

Type = String; **minLength** = N/A; **maxLength** = 9999

Parent Elements:

[networkField](#), [networkSubField](#)

Attributes:

None

Child Elements:

None

4.173 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 Basic Fraud toolkit 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](#)

NOTE: Although included in the schema and shown in the example below, the `<chargeback>` element is not supported. To override the chargeback filter for a selected transaction, use the **fraudFilterOverride** flag (see [fraudFilterOverride](#) on page 538). Please consult your Relationship Manager for additional information.

Example: filtering Structure

```
<filtering>
  <prepaid>true or false</prepaid>
  <international>false</international>
  <chargeback>false</chargeback>
</filtering>
```

4.174 finalPayment

The `finalPayment` element is a required child of the `littleInternalRecurringRequestType`. A value of **true** indicates that this is the final payment of the subscription. Since this element is used only in internally generated transaction, you do not need to code for its use.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[littleInternalRecurringRequest](#)

Attributes:

None

Child Elements:

None

4.175 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

4.176 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 Vantiv before processing this transaction type. In some instances, using a Force Capture transaction can lead to chargebacks and fines.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [amount](#), [orderSource](#), choice of [card](#), [token](#), [mpos](#), [paypage](#), or [applepay](#)

Optional: [billToAddress](#), [customBilling](#), [taxType](#), [enhancedData](#), [processingInstructions](#), [pos](#), [amexAggregatorData](#), [merchantData](#), [productCode](#), [secondaryAmount](#), [surchargeAmount](#), [debtRepayment](#), [processingType](#)

4.177 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
<code>id</code>	String	Yes	The response returns the same value submitted in the Force Capture transaction. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	The response returns the same value submitted in the Force Capture transaction. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	The response returns the same value submitted in the Force Capture transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [tokenResponse](#), [accountUpdater](#), [giftCardResponse](#), [applepayResponse](#)

NOTE: The `postDate` child element is returned only in responses to Online transactions.

4.178 formatId

The `formatId` element is a required child of the `mpos` element. This element identifies the format of the encrypted track submitted from the device.

Type = String; **minLength** = 1; **maxLength** = 1028

Parent Elements:

[mpos](#)

Attributes:

None

Child Elements:

None

4.179 fraudCheck

The `fraudCheck` element is the parent element for the standalone Advanced Fraud Check transaction. You use this transaction type to retrieve the results of the Advanced Fraud Check tool without submitting an Authorization or Sale transaction. You can use this element only in Online transactions.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [advancedFraudChecks](#)

Optional: [billToAddress](#), [shipToAddress](#), [amount](#)

4.180 fraudCheckResponse

The `fraudCheckResponse` element is the parent element for information returned to you in response to a standalone Fraud Check transaction.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [advancedFraudResults](#)

4.181 fraudFilterOverride

The `fraudFilterOverride` element is an optional child of both the `authorization` and the `sale` elements. A setting of **true** will override (disable) all fraud filters for the submitted transaction.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

None

4.182 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:

`activateResponse`, `authorizationResponse`, `deactivateResponse`, `loadResponse`, `saleResponse`,
`unloadResponse`, `authInformation`

Attributes:

None

Child Elements: (All Optional)

`authenticationResult`, `avsResult`, `cardValidationResult`, `advancedAVSResult`,
`advancedFraudResults`

Example: fraudResult Structure

```
<fraudResult>
    <avsResult>00</avsResult>
    <cardValidationResult>N</cardValidationResult>
    <authenticationResult>2</authenticationResult>
    <advancedAVSResult>011</advancedAVSResult>
    <advancedFraudResults>
        <deviceReviewStatus>pass, fail, review, or unavailable</deviceReviewStatus>
        <deviceReputationScore>Score from ThreatMetix</deviceReputationScore>
        <triggeredRule>Triggered Rule_may occur multiple times</triggeredRule>
    </advancedFraudResults>
</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 be enabled specifically to use the Advanced AVS feature. Please consult your Relationship Manager for additional information.

4.183 fundingInstructionVoid

The `fundingInstructionVoid` element is the parent element for the transaction type that a PayFac uses to void a submitted, but not yet processed funding instruction. You must submit the `fundingInstructionVoid` transaction prior to your cutoff time on the same day you submitted the instruction to be voided. This rule also applies to funding instructions submitted on weekends.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#)

4.184 fundingInstructionVoidResponse

The `fundingInstructionVoidResponse` element is the parent element for information returned to you in response to a `fundingInstructionVoid` transaction.

Parent Elements:

[batchResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	The response returns the same value submitted in the credit transaction. minLength = 1 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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

4.185 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, Credit, Debit, or FSA and if Prepaid, the available balance, the type of prepaid card, and whether it is reloadable.

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 Insights features (see [Customer Insight Features](#) on page 25.)

Please consult your Relationship Manager for additional information.

4.186 fundingSubmerchantId

The `fundingSubmerchantId` element is a required child of each funding instruction request transaction type, where it specifies the identifier of the submerchant whose funds are moved by the instruction.

NOTE: If you are processing solely on the Vantiv Core platform or on both the Vantiv Core and Vantiv eCommerce platforms, the value of the `fundingSubmerchantId` is the Vantiv supplied Sub-merchant Merchant Id.

If you are processing solely on the Vantiv eCommerce platform, the value of the `fundingSubmerchantId` is the value of `fundingSubmerchantId` the returned in the Sub-merchant Retrieval Request. Please refer to the *Vantiv PayFac API Reference Guide* for additional information.

Type = String; `minLength` = N/A; `maxLength` = 50

Parent Elements:

[payFacCredit](#), [payFacDebit](#), [physicalCheckCredit](#), [physicalCheckDebit](#), [reserveCredit](#), [reserveDebit](#), [submerchantCredit](#), [submerchantDebit](#), [vendorCredit](#), [vendorDebit](#)

Attributes:

None

Child Elements:

None

4.187 fundsTransferId

The fundingTransferId element is a required child of each funding instruction request/response transaction type, where it specifies the PayFac assigned identifier for the transaction. You must use unique values for each transaction across your entire organization. This identifier is mirrored in the response messages.

Type = String; minLength = N/A; maxLength = 36

Parent Elements:

[payFacCredit](#), [payFacCreditResponse](#), [payFacDebit](#), [payFacDebitResponse](#), [physicalCheckCredit](#), [physicalCheckCreditResponse](#), [physicalCheckDebit](#), [physicalCheckDebitResponse](#), [reserveCredit](#), [reserveCreditResponse](#), [reserveDebit](#), [reserveDebitResponse](#), [submerchantCredit](#), [submerchantCreditResponse](#), [submerchantDebit](#), [submerchantDebitResponse](#), [vendorCredit](#), [vendorCreditResponse](#), [vendorDebit](#), [vendorDebitResponse](#)

Attributes:

None

Child Elements:

None

4.188 giftCardAuthReversal

The `giftCardAuthReversal` element is the parent element for the transaction type that reverses an authorization on a Gift Card.

Parent Elements:

[littleOnlineRequest](#), [littleRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (Required)

[card](#), [originalRefCode](#), [originalAmount](#), [originalTxnTime](#), [originalSystemTraceId](#), [originalSequenceNumber](#)

Child Elements: (Optional)

[littleTxnId](#)

4.189 giftCardAuthReversalResponse

The `giftCardAuthReversalResponse` element is the parent element for information returned to you in response to a Gift Card Auth Reversal transaction.

Parent Elements:

[littleOnlineResponse](#), [littleResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#), [giftCardResponse](#)

Optional: [postDate](#) (returned for online transactions only)

4.190 giftCardBin

The `giftCardBin` element is a required child of the `virtualGiftCard` element defining the requested BIN of the virtual Gift Card number requested.

NOTE: In an early iteration of schema V8.22 issued in September of 2013 this element was defined as an optional child of the `virtualGiftCard` element. If you coded to the earlier version, be aware that this element is now required. If you do not include this element, the transaction will fail XML validation.

Type = String; **minLength** = N/A; **maxLength** = 10

Parent Elements:

[virtualGiftCard](#)

Attributes:

None

Child Elements:

None

4.191 giftCardCapture

The `giftCardCapture` element is the parent element for the transaction type that captures funds previously authorized on a Gift Card.

Parent Elements:

[littleOnlineRequest](#), [littleRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (Required)

[card](#), [captureAmount](#), [originalRefCode](#), [originalAmount](#), [originalTxnTime](#),

Child Elements: (Optional)

[littleTxnId](#)

4.192 giftCardCaptureResponse

The `giftCardCaptureResponse` element is the parent element for information returned to you in response to a Gift Card Capture transaction.

Parent Elements:

[littleOnlineResponse](#), [littleResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	The response returns the same value submitted in the Force Capture transaction. minLength = 1 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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#), [giftCardResponse](#)

Optional: [fraudResult](#), [postDate](#) (returned for online transactions only)

4.193 giftCardCredit

The `giftCardCredit` element is the parent element for all Gift Card Credit transactions. There are two possible structures for this message type. You use the first structure for credit transactions against captures/sales processed by Vantiv. The alternate structure applies to credits against captures/sales not processed by Vantiv. You can use this transaction type in either Online or Batch submissions.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

For credits to transactions processed by Vantiv (Required): [creditAmount](#), [card](#)

For credits to transactions processed by Vantiv (Optional): [littleTxnId](#)

For credits to transactions not processed by Vantiv (Required): [orderId](#), [creditAmount](#), [orderSource](#), [card](#)

4.194 giftCardCreditResponse

The `giftCardCreditResponse` element is the parent element for information returned to you in response to a Gift Card Credit transaction.

Parent Elements:

[littleOnlineResponse](#), [littleResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#), [giftCardResponse](#)

Optional: [fraudResult](#), [postDate](#) (returned for online transactions only)

4.195 giftCardResponse

The `giftCardResponse` element is a required child of several transaction types and optional for several others. Through its child elements, it provides details about the available balance on a Gift Card, as well as additional reference items used in subsequent, related transactions. For example, in a `giftCardAuthReversal` transaction, you must include values of the `refCode`, `systemTraceId`, and `sequenceNumber` from the `authorizationResponse` in the `originalRefCode`, `originalSystemTraceId`, and `originalSequenceNumber` elements respectively.

Parent Elements:

Required:[activateResponse](#), [activateReversalResponse](#), [giftCardAuthReversalResponse](#), [balanceInquiryResponse](#), [giftCardCaptureResponse](#), [giftCardCreditResponse](#), [deactivateResponse](#), [deactivateReversalResponse](#), [depositReversalResponse](#), [loadResponse](#), [loadReversalResponse](#), [refundReversalResponse](#), [unloadResponse](#), [unloadReversalResponse](#)

Optional (per schema, but always returned for Gift Card Transactions): [authorizationResponse](#), [captureGivenAuthResponse](#), [forceCaptureResponse](#), [saleResponse](#)

Attributes:

None

Child Elements (All Optional, but must contain at least one child):

Optional: `txnTime`, `refCode`, `systemTraceId`, `sequenceNumber`, `availableBalance`, `beginningBalance`, `endingBalance`, `cashBackAmount`

NOTE:

Although included in the schema, the `beginningBalance`, `endingBalance`, and `cashBackAmount` elements are not currently supported.

Example: giftCardResponse Structure

```
<giftCardResponse>
  <txnTime>2016-11-15T12:00:00</txnTime>
  <refCode>abc123</refCode>
  <systemTraceId>123456</systemTraceId>
  <sequenceNumber>123456</sequenceNumber>
  <availableBalance>Balance Available on Gift Card</availableBalance>
</giftCardResponse>
```

4.196 giropay

The `giropay` element is a child of the `sale` transaction that, through its child elements, defines information needed to process an Giropay (Real-time Bank Transfer) transaction. At this time, you can use the iDeal method of payment in Online transactions only.

Parent Elements:

[sale](#)

Attributes:

None

Child Elements:

Optional: [preferredLanguage](#)

Example: giropay Structure

```
<giropay>
    <preferredLanguage>Country Code of Language</preferredLanguage>
</giropay>
```

4.197 giropayResponse

The `giropayResponse` element is a child of the `saleResponse` transaction when the method of payment in the request is `giropay`. It contains child elements that you should store for future use/reference.

Parent Elements:

[saleResponse](#)

Attributes:

None

Child Elements (all Optional):

[paymentPurpose](#), [redirectUrl](#), [redirectToken](#)

Example: giropayResponse Structure

```
<giropayResponse>
    <redirectUrl>URL of Vantiv Supplied Mandate</redirectUrl>
    <redirectToken>Map Mandate to littleTxnId</redirectToken>
    <paymentPurpose>Reference Number + Billing Descriptor</paymentPurpose>
</giropayResponse>
```

4.198 header

The `header` element is a required child of the `applepay` element. Its child elements provides information required to process the Apple Pay transaction.

Type = String; **minLength** = N/A; **maxLength** = 10000

Parent Elements:

[applepay](#)

Attributes:

None

Child Elements (All Optional, but must contain at least one child):

Optional: [applicationData](#), [ephemeralPublicKey](#), [publicKeyHash](#), [transactionId](#)

Example: header Structure

```
<header>Password</header>
    <applicationData>SHA-256 Hash Hex Encoded of App Data
Property</applicationData>
    <ephemeralPublicKey>Base64 Encoded Ephemeral Public Key</ephemeralPublicKey>
    <publicKeyHash>Base64 Hash of Public Key Bytes from Merchant
Certttificate</publicKeyHash>
    <transactionId>Hex Transaction Id</transactionId>
</header>
```

4.199 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>
```

4.200 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>
```

4.201 iban

The `iban` element is a required child of the `sepaDirectDebit` element. It defines the International Bank Account Number of the customer.

Type = String; **minLength** = 15; **maxLength** = 34

Parent Elements:

[sepaDirectDebit](#)

Attributes:

None

Child Elements:

None

4.202 ideal

The `ideal` element is a child of the `sale` transaction that, through its child elements, defines information needed to process an iDEAL (Real-time Bank Transfers) transaction. At this time, you can use the iDEAL method of payment in Online transactions only.

Parent Elements:

[sale](#)

Attributes:

None

Child Elements:

Optional: [preferredLanguage](#)

Example: ideal Structure

```
<ideal>
    <preferredLanguage>Country Code of Language</preferredLanguage>
</ideal>
```

4.203 idealResponse

The `idealResponse` element is a child of the `saleResponse` transaction when the method of payment in the request is `ideal`. It contains child elements that you should store for future use/reference.

Parent Elements:

[saleResponse](#)

Attributes:

None

Child Elements (all Optional):

[paymentPurpose](#), [redirectUrl](#), [redirectToken](#)

Example: `idealResponse` Structure

```
<idealResponse>
    <redirectUrl>URL of Vantiv Supplied Mandate</redirectUrl>
    <redirectToken>Map Mandate to littleTxnId</redirectToken>
    <paymentPurpose>Reference Number + Billing Descriptor</paymentPurpose>
</idealResponse>
```

4.204 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

4.205 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 PayPal Credit transactions.

NOTE:

As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = Long; **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.206 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 PayPal Credit transactions.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

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

Enumeration	Description
SEK	Swedish Kronor
SGD	Singapore Dollar
USD (default)	United States Dollar

4.207 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

4.208 intervalType

The **intervalType** element is a required child of the **createPlan** element and defines the billing period associated with the Plan.

Type = String (enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[createPlan](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
ANNUAL	The billing period is once per year.
SEMIANNUAL	The billing period is twice per year.
QUARTERLY	The billing period is every three months.
MONTHLY	The billing period is every month.
WEEKLY	The billing period is every week.

4.209 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

4.210 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

4.211 itemCategoryCode

The `itemCategoryCode` element is an optional child of the `billMeLaterRequest` element and defines the PayPal Credit item category for the type of product sold.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = Integer; **totalDigits** = 4

Parent Elements:

[billMeLaterRequest](#)

Attributes:

None

Child Elements:

None

4.212 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

4.213 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

4.214 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

4.215 ksn

The `ksn` element is a required child of the `mpos` element. This element defines the Key serial Number returned from the encrypting device. It is created automatically from the unique identifier of the device and an internal transaction counter.

Type = String; **minLength** = 1; **maxLength** = 1028

Parent Elements:

`mpos`

Attributes:

None

Child Elements:

None

4.216 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

4.217 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>
```

4.218 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

4.219 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

4.220 littleInternalRecurringRequest

The littleInternalRecurringRequest element and its children is an element structure that exists solely for internally (to system) generated transactions associated with recurring payments managed by the Recurring engine. You do not need to code for this structure.

Parent Elements:

[sale](#)

Attributes:

None

Child Elements:

[subscriptionId](#), [recurringTxnId](#), [finalPayment](#)

4.221 littleOnlineRequest

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.little.com/schema. minLength = N/A maxLength = 38
merchantId	String	Yes	A unique string used to identify the merchant within the system. minLength = N/A maxLength = 50 Note: International currencies are supported on a per merchantId basis.
loggedInUser	String	No	Internal Use Only

Child Elements:

Required: [authentication](#)

One of the following required: [activate](#), [activateReversal](#), [authorization](#), [authReversal](#), [balanceInquiry](#), [cancelSubscription](#), [capture](#), [captureGivenAuth](#), [createPlan](#), [credit](#), [deactivate](#), [deactivateReversal](#), [depositReversal](#), [echeckCredit](#), [echeckRedeposit](#), [echeckSale](#), [echeckVerification](#), [echeckVoid](#), [forceCapture](#), [fraudCheck](#), [giftCardAuthReversal](#), [giftCardCapture](#), [giftCardCredit](#), [load](#), [loadReversal](#), [registerTokenRequest](#), [refundReversal](#), [sale](#), [unload](#), [updateCardValidationNumOnToken](#), [updatePlan](#), [updateSubscription](#), [unloadReversal](#), [void](#)

4.222 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. 2 - Indicates that the submitted content was either improperly formatted XML or non-XML content. 3 - Indicates that the submission contains empty or invalid credentials (user and password). 4 - Indicates that the merchant has reached the maximum number of concurrent connections. 5 - Indicates that systems may have detected message content that violates certain restrictions. minLength = N/A maxLength = 3

Attribute Name	Type	Required?	Description
message	String	Yes	<p>XML validation error message. Expected values are as follows:</p> <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 859 for example messages. • If the response attribute returns 2, the message attribute is "System Error - Call Little & Co." • If the response attribute returns a value of 3, 4, or 5, the message attribute is "There is a problem with the system. Contact eCommerceSupport@vantiv.com." <p>minLength = N/A maxLength = 512</p>

Child Elements:

One of the following required: [activateResponse](#), [activateReversalResponse](#), [authorizationResponse](#), [authReversalResponse](#), [balanceInquiryResponse](#), [cancelSubscriptionResponse](#), [captureGivenAuthResponse](#), [captureResponse](#), [createPlanResponse](#), [deactivateResponse](#), [deactivateReversalResponse](#), [depositReversalResponse](#), [creditResponse](#), [echeckCreditResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [forceCaptureResponse](#), [giftCardAuthReversalResponse](#), [giftCardCaptureResponse](#), [giftCardCreditResponse](#), [loadResponse](#), [loadReversalResponse](#), [refundReversalResponse](#), [registerTokenResponse](#), [saleResponse](#), [unloadResponse](#), [unloadReversalResponse](#), [updateCardValidationNumOnTokenResponse](#), [voidResponse](#), [updatePlanResponse](#), [updateSubscriptionResponse](#)

4.223 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: http://www.little.com/schema. minLength = N/A maxLength = 38
id	String	No	A unique string to identify the session within the system. minLength = N/A maxLength = 25
numBatchRequests	Integer	Yes	Defines the total number of batchRequest children included in the littleRequest. If the littleRequest contains only an RFRRequest, then set this attribute to "0".

Child Elements:

Required: [authentication](#)

One of the following required: [batchRequest](#), [RFRRequest](#)

4.224 littleResponse

This is the root element for all LittleXML Batch 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
id	String	No	The response returns the same value submitted in the Little Request. 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. 2 - Indicates that the submitted content was either improperly formatted XML or non-XML content. 3 - Indicates that the submission contains empty or invalid credentials (user and password). 4 - Indicates that the merchant has reached the maximum number of concurrent connections. 5 - Indicates that systems may have detected message content that violates certain restrictions. minLength = N/A maxLength = 3

Attribute Name	Type	Required?	Description
message	String	Yes	<p>XML validation error message. Expected values are as follows:</p> <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 859 for example messages. • If the response attribute returns 2, the message attribute is "System Error - Call Little & Co." • If the response attribute returns a value of 3, 4, or 5, the message attribute is "There is a problem with the system. Contact eCommerceSupport@vantiv.com." <p>minLength = N/A maxLength = 512</p>
littleSessionId	Long	Yes	<p>A unique value assigned by Vantiv to identify the session.</p> <p>minLength = N/A maxLength = 19</p>

Child Elements:

One of the following required: [batchResponse](#), [RFRResponse](#)

4.225 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

4.226 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

4.227 littleTxnId

The `littleTxnId` element is used to identify transactions in the 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](#), [activateResponse](#), [activateReversalResponse](#), [authorizationResponse](#), [authReversalResponse](#), [capture](#), [captureResponse](#), [credit](#), [creditResponse](#), [captureGivenAuthResponse](#), [deactivateResponse](#), [deactivateReversalResponse](#), [depositReversalResponse](#), [echeckCredit](#), [echeckCreditResponse](#), [echeckPreNoteCreditResponse](#), [echeckPreNoteSaleResponse](#), [echeckRedeposit](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [echeckVoid](#), [echeckVoidResponse](#), [forceCapture](#), [forceCaptureResponse](#), [fraudCheckResponse](#), [fundingInstructionVoid](#), [giftCardAuthReversalResponse](#), [giftCardCaptureResponse](#), [giftCardCreditResponse](#), [loadResponse](#), [loadReversalResponse](#), [payFacCreditResponse](#), [payFacDebitResponse](#), [physicalCheckCreditResponse](#), [physicalCheckDebitResponse](#), [refundReversalResponse](#), [saleResponse](#), [unloadResponse](#), [void](#), [voidResponse](#), [cancelSubscriptionResponse](#), [updatePlanResponse](#), [updateSubscriptionResponse](#), [unloadReversalResponse](#), [submerchantCreditResponse](#), [submerchantDebitResponse](#), [vendorCreditResponse](#), [vendorDebitResponse](#)

NOTE: While the `littleTxnId` is optional in the transaction types listed below, Vantiv recommends you include it whenever possible. Failure to include the `littleTxnId` results in missing information in the Associated Transaction Stream section of the Transaction Detail screen in iQ.

This element is an optional child of the following: [activateReversal](#), [deactivateReversal](#), [depositReversal](#), [giftCardAuthReversal](#), [giftCardCapture](#), [giftCardCredit](#), [loadReversal](#), [refundReversal](#), [unloadReversal](#)

NOTE: Although the schema shows the `littleTxnId` element as an optional child of the authorization, `echeckSale`, and `sale` transactions, under normal circumstances, merchants would never use this option.

Attributes:

None

Child Elements:

None

4.228 load

The `load` element is the parent element for the transaction type that adds funds to a reloadable Gift Card.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (all Required)

[orderId](#), [amount](#), [orderSource](#), [card](#)

4.229 loadResponse

The `loadResponse` element is the parent element for information returned to you in response to a `load` 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
<code>id</code>	String	Yes	The response returns the same value submitted in the Load transaction. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	The response returns the same value submitted in the Load transaction. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	The response returns the same value submitted in the Load transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [fraudResult](#), [giftCardResponse](#)

4.230 loadReversal

The `loadReversal` element is the parent element for the transaction type that reverses the loading of a Gift Card.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (Required)

[card](#), [originalRefCode](#), [originalAmount](#), [originalTxnTime](#), [originalSystemTraceId](#), [originalSequenceNumber](#)

Child Elements: (Optional)

[littleTxnId](#)

4.231 loadReversalResponse

The `loadReversalResponse` element is the parent element for information returned to you in response to an `loadReversal` transaction.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [giftCardResponse](#)

4.232 mandateProvider

The `mandateProvider` element is a required child of the `sepaDirectDebit` element and defines whether the Merchant or Vantiv provides the mandate for customer approval.

Type = String (Enum); **Allowed Values** = Merchant or Vantiv

Parent Elements:

[sepaDirectDebit](#)

Attributes:

None

Child Elements:

None

4.233 mandateReference

The `mandateReference` element is an optional child of both the `sepaDirectDebit` element and the `sepaDirectDebitResponse` element. You use this element for recurring payments (after the initial transaction) to provide the reference number that links subsequent payments in a recurring stream to the mandate agreed to at the time of the initial payment. Vantiv returns this value in the `sepaDirectDebitResponse` associated with the initial payment. It is not returned for subsequent payments in a recurring stream.

Type = String; **minLength** = N/A; **maxLength** = 256

Parent Elements:

[sepaDirectDebit](#), [sepaDirectDebitResponse](#)

Attributes:

None

Child Elements:

None

4.234 mandateSignatureDate

The `mandateSignatureDate` element is an optional child of the `sepaDirectDebit` element and defines the date the consumer agreed to the mandate allowing the merchant to debit their account. Although optional, you should always provide this information when the value for the `mandateProvider` element is Merchant.

Type = Date; **Format** = YYYY-MM-DD

Parent Elements:

[sepaDirectDebit](#)

Attributes:

None

Child Elements:

None

4.235 mandateURL

The `mandateUrl` element is an optional child of the `sepaDirectDebit` element and defines the URL of the mandate to which the consumer agreed, allowing the merchant to debit their account. Although optional, you should always provide this information when the value for the `mandateProvider` element is Merchant.

Type = String; minLength = 1; maxLength = 2000

Parent Elements:

[sepaDirectDebit](#)

Attributes:

None

Child Elements:

None

4.236 matchCount

The matchCount element is a required child of the queryTransactionResponse element and defines the number of found transactions that matched the criteria submitted in the queryTransaction.

Type = Integer; **totalDigits** = 2

Parent Elements:

[queryTransactionResponse](#)

Attributes:

None

Child Elements:

None

4.237 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.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [echeckCredit](#), [echeckPreNoteCredit](#), [echeckPreNoteSale](#), [echeckRedeposit](#), [echeckSale](#), [echeckVerification](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements (all optional):

[affiliate](#), [campaign](#), [merchantGroupId](#)

4.238 merchantGroupId

The `merchantGroupId` element is an optional child element of the `merchantData` element. You can use it to track transactions based upon this user defined parameter.

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

Parent Elements:

[merchantData](#)

Attributes:

None

Child Elements:

None

4.239 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 system.

Type = String; **minLength** = N/A; **maxLength** = 50

Parent Elements:

[accountUpdateFileRequestData](#)

Attributes:

None

Child Elements:

None

NOTE:

Several elements use `merchantId` as an attribute, including `batchRequest`, `batchResponse`, and `littleOnlineRequest`.

4.240 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 response codes and associated messages, please refer to [Appendix A](#).

Type = String; **minLength** = N/A; **maxLength** = 512

Parent Elements:

activateResponse, activateReversalResponse, authorizationResponse, captureResponse, captureGivenAuthResponse, creditResponse, deactivateResponse, deactivateReversalResponse, depositReversalResponse, echeckCreditResponse, echeckPreNoteCreditResponse, echeckPreNoteSaleResponse, echeckRedepositResponse, echeckSalesResponse, echeckVerificationResponse, echeckVoidResponse, extendedCardResponse, forceCaptureResponse, fraudCheckResponse, loadResponse, loadReversalResponse, queryTransactionResponse, queryTransactionUnavailableResponse, refundReversalResponse, saleResponse, unloadResponse, unloadReversalResponse, voidResponse, cancelSubscriptionResponse, updatePlanResponse, updateSubscriptionResponse, payFacCreditResponse, payFacDebitResponse, physicalCheckCreditResponse, physicalCheckDebitResponse, reserveCreditResponse, reserveDebitResponse, submerchantCreditResponse, submerchantDebitResponse, vendorCreditResponse, vendorDebitResponse

Attributes:

None

Child Elements:

None

4.241 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

4.242 mpos

The `mpos` element defines payment card information when the transaction originates with a ROAM device.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#), [registerTokenRequest](#)

Attributes:

None

Child Elements: (all Required)

[ksn](#), [formatId](#), [encryptedTrack](#), [track1Status](#), [track2Status](#)

Example: mpos Structure

```
<mpos>
  <ksn>Key Serial Number</ksn>
  <formatId>Format of Encrypted Track</formatId>
  <encryptedTrack>Encrypted Track Data</encryptedTrack>
  <track1Status>Card Validation Number</track1Status>
  <track2Status>Card Validation Number</track2Status>
</mpos>
```

4.243 name

The name element defines the customer name in both the billToAddress and shipToAddress elements. When used as a child of one of the Recurring Engine associated parents (i.e., createAddOn, updateAddOn, createDiscount, updateDiscount, or createPlan), the name element specifies the name of the parent item being created/updated.

Type = String; minLength = N/A; maxLength = 100

Parent Elements:

[billToAddress](#), [shipToAddress](#), [createAddOn](#), [updateAddOn](#), [createDiscount](#), [updateDiscount](#), [createPlan](#)

NOTE:	<p>The name element is required for Echeck transactions. If you do not submit the customer name in an Echeck transaction, we return Response Code 330 - Invalid Payment Type.</p> <p>The name element is required for sale transactions using the sepaDirectDebit method of payment and must be a minimum of two characters. Failure to include the name element results in a 901 - Missing Name Response Code. If the value is not at least two characters, the transaction declines with a response code of 902 - Invalid Name</p>
--------------	--

Attributes:

None

Child Elements:

None

4.244 networkField

The `networkField` element is an optional child of the `networkResponse` element. Its attributes and child elements define the Field Number, Field Name, (Raw) Field Value, as well as any Sub-fields, if applicable.

Parent Elements:

[networkResponse](#)

Attributes:

Attribute Name	Type	Required?	Description
fieldName	String	Yes	<p>This attribute defines the name of the ISO 8583 field containing the information returned.</p> <p>Type = Enum</p> <p>Allowed Values:</p> <ul style="list-style-type: none"> • Transaction Amount • Settlement Amount • Cardholder Billing Amount • Settlement Conversion Rate • Cardholder Billing Conversion Rate • Settlement Date • Authorization Identification Response • Response Code • Additional Response Data • Private Use Additional Data • Settlement Currency Code • Cardholder Billing Currency Code • Additional Amounts • Reserved Private • Transaction Description • Reserved For National Use • Reserved For Private Use
fieldNumber	Integer	Yes	<p>This attribute defines the number of the ISO 8583 field containing the information returned. Different card networks may use different Field Numbers for the same information.</p> <p>minLength = 1 maxLength = N/A</p>

Child Elements:[fieldValue](#), [networkSubField](#)**Example: networkField Structure with fieldValue**

```
<networkField fieldNumber="4" fieldName="Transaction Amount">
    <fieldValue>000000001111</fieldValue>
</networkField>
```

Example: networkField Structure with networkSubField

```
<networkField fieldNumber="44" fieldName="Additional Response Data">
    <networkSubField fieldNumber="1">
        <fieldValue>7</fieldValue>
    </networkSubField>
    <networkSubField fieldNumber="2">
        <fieldValue>W</fieldValue>
    </networkSubField>
</networkField>
```

4.245 networkResponse

The `networkResponse` element is an optional child of the `enhancedAuthResponse` element. Its child elements provide a number of data points returned by the card networks in their ISO 8583 response messages.

Parent Elements:

[enhancedAuthResponse](#)

Attributes:

None

Child Elements:

[endpoint](#), [networkField](#)

Example: networkResponse Structure

```
<networkResponse>
    <endpoint>VISA</endpoint>
    <networkField fieldNumber="4" fieldName="Transaction Amount">
        <fieldValue>00000000962</fieldValue>
    </networkField>
    <networkField fieldNumber="44" fieldName="Additional Response Data">
        <networkSubField fieldNumber="1">
            <fieldValue>7</fieldValue>
        </networkSubField>
        <networkSubField fieldNumber="2">
            <fieldValue>W</fieldValue>
        </networkSubField>
    </networkField>
</networkResponse>
```

4.246 networkSubField

The `networkSubField` element is a required child of the `networkField` element, when a subfield exists. Its child element provides the raw subfield data returned by the card networks in their ISO 8583 response messages.

Parent Elements:

[networkField](#)

Attributes:

Attribute Name	Type	Required?	Description
fieldNumber	Integer	Yes	This attribute defines the number of the ISO 8583 field or subfield containing the information returned. Different card networks may use different Field Numbers for the same information. minLength = 1 maxLength = N/A

Child Elements:

[fieldValue](#)

4.247 networkTransactionId

The `networkTransactionId` element is an optional child of the `authorizationResponse`, and `saleResponse` transactions returned for Visa and Discover transactions. Visa and Discover use this value to link subsequent payments in a recurring stream back to the initial transaction, when the initial transaction involves network tokens, such as Apple Pay or Android Pay transactions. You must include this value in the request message (`originalNetworkTransactionId` element) for subsequent recurring payments, if the payment method is either Visa or Discover.

Type = String; **minLength** = N/A; **maxLength** = 30

NOTE: Currently, the system returns the value `networkTransactionId` only for Visa transactions. A future release will include support for Discover.

As of April 2017, if you fail to include this value in the element for a recurring Visa or Discover transaction involving a network token, the card network will reject the transaction.

Parent Elements:

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

Attributes:

None

Child Elements:

None

4.248 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>
```

4.249 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>
```

4.250 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>
```

4.251 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, litleToken, routingNum

Example: newAccountInfo Structure

```
<newTokenInfo>
    <accType>Account Type</accType>
    <litleToken>New Token Number</litleToken>
    <routingNum>New Routing Number</routingNum>
</newTokenInfo>
```

4.252 nextRecycleTime

NOTE: The recycling Advice feature is no longer supported.

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

4.253 number

The `number` element defines the account number associated with the transaction or the new/old account number associated with an update. This is a required child of the `card` element for card-not-present transactions.

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

Parent Elements:

[accountInformation](#), [card](#), [newCardInfo](#), [originalCardInfo](#)

Attributes:

None

Child Elements:

None

4.254 **numberOfPayments**

The `numberOfPayments` element defines the number of payments in a recurring billing plan including the initial payment. The timing of subsequent charges is defined by the `planCode` element. This element is an optional child of both the `subscription`, and `createPlan` elements. When submitted as a child of the `subscription` element, the value overrides the default value defined in the Plan.

NOTE: For an open-ended Plan, please omit the optional `numberOfPayments` element in `createPlan`.

For an open-ended subscription, please omit the optional `numberOfPayments` element in `subscription` and reference an open-ended Plan.

Type = Integer; **minLength** = 1; **maxLength** = 99

Parent Elements:

[subscription](#), [createPlan](#)

Attributes:

None

Child Elements:

None

4.255 onlinePaymentCryptogram

The `onlinePaymentCryptogram` element is an optional child of the `applepayResponse` element and provides the BASE64 Encoded signature cryptogram associated with the Apple Pay transaction.

Type = Base 64 Encoded String; **minLength** = N/A; **maxLength** = 56

Parent Elements:

[applepayResponse](#)

Attributes:

None

Child Elements:

None

4.256 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

4.257 orderId

The orderId element defines a merchant-assigned value representing the order in the merchant's system.

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

NOTE: If you are using the orderId element as the transaction signature for the Recycling Engine, do not use the pipe character ("|") in the orderId. Use of the pipe character in this scenario will cause recycling errors.

Parent Elements:

accountUpdate, activate, authorization, authorizationResponse, balanceInquiry, credit, captureGivenAuth, deactivate, echeckCredit, echeckPreNoteCredit, echeckPreNoteSale, echeckSale, echeckVerification, forceCapture, giftCardCredit, load, registerTokenRequest, sale, saleResponse, unload, updateCardValidationNumOnToken

Attributes:

None

Child Elements:

None

4.258 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:

activate, authorization, balanceInquiry, captureGivenAuth, credit, deactivate, echeckCredit, echeckPreNoteCredit, echeckPreNoteSale, echeckSale, echeckVerification forceCapture, load, sale, unload

Attributes:

None

Child Elements:

None

Enumerations:

NOTE:	If you submit the wrong <code>orderSource</code> value, we return the response code 370 - Internal System Error - Contact Little. eCheckSale transactions must use an <code>orderSource</code> of one of the following: telephone, ecommerce, echeckppd, or recurringtel.
--------------	--

Enumeration	Description
3dsAuthenticated	Use this value only if you authenticated the cardholder via an approved 3DS system such as Visa Verified By Visa and MasterCard SecureCode. This value applies to Visa and MasterCard transactions only. NOTE: Your Merchant Profile must be configured to process 3DS type payments and accept this value.
3dsAttempted	Use this value only if you attempted to authenticate the cardholder via an approved 3DS system such as Visa VerifiedByVisa and MasterCard SecureCode, but either the Issuer or cardholder is not participating in the 3DS program. This value applies to Visa and MasterCard transactions only. If this is a MasterCard transaction, you must include the <code>authenticationValue</code> returned by MasterCard. NOTE: Your Merchant Profile must be configured to process 3DS type payments and accept this value.

Enumeration	Description
echeckppd	(eCheck only) Use this value for eCheck PPD transactions (Prearranged Payment and Deposit Entries). This type of transaction occurs when a merchant receives a written authorization, including a voided paper check, from a consumer so that the merchant can debit the consumer account. These transactions can be single entry or recurring debits to a consumer's account.
ecommerce	The transaction is an Internet or electronic commerce transaction.
installment	The transaction is an installment payment.
mailorder	The transaction is for a single mail order transaction.
recurring	The transaction is a recurring transaction. For Visa transactions, you can use this value for all transactions in a recurring stream including the initial 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
applepay	The transaction uses the Apple Pay service.
androidpay	The transaction uses the Android Pay service.

4.259 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>
```

4.260 origAccountNumber

The `origAccountNumber` element is an optional child of the `queryTransaction` element and defines the account number of the credit/debit/gift card used in the original transaction.

NOTE: If you are performing a query for an eCheck transaction, do not use this element to designate the checking/savings account number. You should use this element for Credit/Debit/Gift card account numbers only.

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

Parent Elements:

[queryTransaction](#)

Attributes:

None

Child Elements:

None

4.261 origActionType

The `origActionType` element is a required child of the `queryTransaction` element and defines the transaction type of original transaction.

Type = String (Enum); **minLength** = 1; **maxLength** = 2 (Valid values shown below)

Parent Elements:

[queryTransaction](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Transaction Type
A	authorization
AR	activateReversal
D	deposit or sale
G	activate
I	unload
J	deactivate
L	load
LR	loadReversal
P	authReversal
R	credit
RR	refundReversal
S	echeckSale
T	echeckCredit
UR	unloadReversal
V	void
W	depositReversal

Enumeration	Transaction Type
X	echeckVoid

4.262 origId

The `origId` element is a required child of the `queryTransaction` element and defines the `id` attribute used in the original transaction.

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

Parent Elements:

[queryTransaction](#)

Attributes:

None

Child Elements:

None

4.263 originalAmount

The `originalAmount` element is a required child of several gift card reversal transactions, as well as the `giftCardCapture` transaction, where it specifies the amount of the original gift card transaction. In the case of the `giftCardCapture` transaction, this is the amount associated with the authorization. In the case of a reversal transaction, it is the amount from the transaction you want to reverse. For example, for an `loadReversal` it specifies the amount from the `load` transaction.

Type = Integer; **totalDigits** = 6

Parent Elements:

[activateReversal](#), [deactivateReversal](#), [depositReversal](#), [giftCardAuthReversal](#),[giftCardCapture](#),
[loadReversal](#), [refundReversal](#), [unloadReversal](#)

Attributes:

None

Child Elements:

None

4.264 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>
```

4.265 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>
```

4.266 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>
```

4.267 originalNetworkTransactionId

The `originalNetworkTransactionId` element is an optional child of the `authorization`, `captureGivenAuth`, and `sale` transactions. It defines the `networkTransactionId` returned in the response messages for the transaction types previously mentioned. You must include this element and the original value returned for subsequent (after the initial) recurring payments, if the payment method is either Visa or Discover.

Type = String; **minLength** = N/A; **maxLength** = 30

NOTE: Currently, the system returns the value `networkTransactionId` only for Visa transactions. A future release will include support for Discover.
As of April 2017, if you fail to include this element/value for a recurring Visa or Discover transaction, the card network will reject the transaction.

Parent Elements:

`authorization`, `captureGivenAuth`, `sale`

Attributes:

None

Child Elements:

None

4.268 originalRefCode

The `originalRefCode` element is a required child of several gift card transactions, where it specifies the value returned in the `refCode` element of the associated `giftCardResponse` element of the response messages.

Type = String; **minLength** = N/A; **maxLength** = 6

Parent Elements:

[activateReversal](#), [deactivateReversal](#), [depositReversal](#), [giftCardAuthReversal](#), [giftCardCapture](#), [loadReversal](#), [refundReversal](#), [unloadReversal](#),

Attributes:

None

Child Elements:

None

4.269 originalSequenceNumber

The `originalSequenceNumber` element is a required child of several gift card reversal transactions, where it specifies the value returned in the `sequenceNumber` element of the associated `giftCardResponse` element of the response messages.

Type = String; **totalDigits** = 6; **Allowed Characters:** 0 - 9

Parent Elements:

[activateReversal](#), [deactivateReversal](#), [depositReversal](#), [giftCardAuthReversal](#), [loadReversal](#), [refundReversal](#), [unloadReversal](#),

Attributes:

None

Child Elements:

None

4.270 originalSystemTraceId

The `originalSystemTraceId` element is a required child of several gift card reversal transactions, where it specifies the value returned in the `systemTraceId` element of the associated `giftCardResponse` element from the response messages of the transaction you wish to reverse. For example, for a `loadReversal` transaction, you use the value returned in the `loadResponse` message.

Type = Integer; **totalDigits** = 6

Parent Elements:

[activateReversal](#), [deactivateReversal](#), [depositReversal](#), [giftCardAuthReversal](#), [loadReversal](#), [refundReversal](#), [unloadReversal](#)

Attributes:

None

Child Elements:

None

4.271 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>
```

4.272 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>
```

4.273 originalTransactionAmount

The `originalTransactionAmount` element is an optional child of the `authorization`, `captureGivenAuth`, and `sale` transactions. It defines the amount from the initial transaction in a recurring stream. You must include this element and the original amount in subsequent (after the initial) recurring payments, if the payment method is Discover.

Type = Integer; totalDigits = 12

Parent Elements:

[authorization](#), [captureGivenAuth](#), [sale](#)

Attributes:

None

Child Elements:

None

4.274 originalTxnTime

The `originalTxnTime` element is a required child of several gift card transactions, which specifies the date and time the original transaction was processed by Vantiv. The system returns this value in the `txnTime` child of the `giftCardResponse` element. The format of the element is `YYYY-MM-DDTHH:MM:SS`. For example, `2016-11-21T11:00:00`.

Type = `dateTime`; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[activateReversal](#), [deactivateReversal](#), [depositReversal](#), [giftCardAuthReversal](#), [giftCardCapture](#), [loadReversal](#), [refundReversal](#), [unloadReversal](#)

Attributes:

None

Child Elements:

None

4.275 origLittleTxnId

The `origLittleTxnId` element is an optional child of the `queryTransaction` element and defines the value of the `littleTxnId` element assigned to the original transaction and returned in the response message.

Type = Long; **minLength** = N/A; **maxLength** = 19

Parent Elements:

`queryTransaction`

Attributes:

None

Child Elements:

None

4.276 origOrderId

The `origOrderId` element is an optional child of the `queryTransaction` element and defines the merchant-assigned value for the `orderId` element submitted in the original transaction.

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

Parent Elements:

[queryTransaction](#)

Attributes:

None

Child Elements:

None

4.277 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

4.278 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 `<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

4.279 payFacCredit

The `payFacCredit` element is the parent element for the transaction type that a PayFac uses to distribute funds to themselves (i.e., from the PayFac Settlement Account to the PayFac Operating Account).

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#)

4.280 payFacCreditResponse

The `payFacCreditResponse` element is the parent element for information returned to you in response to a `payFacCredit` transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.281 payFacDebit

The `payFacDebit` element is the parent element for the transaction type that a PayFac uses to move funds from the PayFac Operating Account back to the PayFac Settlement Account.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#)

4.282 payFacDebitResponse

The payFacDebitResponse element is the parent element for information returned to you in response to a payFacDebit transaction.

Parent Elements:

[batchRequest](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.283 paymentDataType

The `paymentDataType` element is an optional child of the `applepayResponse` element and specifies data type of the payment data associated with an Apple Pay transaction.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[applepayResponse](#)

Attributes:

None

Child Elements:

None

4.284 paymentPurpose

The `paymentPurpose` element is an optional child of the `idealResponse` and `giropayResponse` elements where it specifies information (equivalent of Bill Descriptor for a credit card transaction) that appears on the consumer bank statement along with a reference string representing the transaction. The value has two components. The first part is a reference number assigned to the transaction. The second part is a default merchant descriptor established during implementation.

Type = String; **minLength** = N/A; **maxLength** = 256

Parent Elements:

[idealResponse](#), [giropayResponse](#)

Attributes:

None

Child Elements:

None

4.285 paypage

The `paypage` element defines eProtect account information. It replaces the `card` or `token` elements in transactions using the eProtect feature of the Vault solution. When you submit the `paypage` element in a request, response messages will include token information.

Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#), [updateSubscription](#)

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 eProtect</paypageRegistrationId>
  <expDate>Expiration Date</expDate>
  <cardValidationNum>Card Validation Number</cardValidationNum>
  <type>Method of Payment</type>
</paypage>
```

4.286 paypageRegistrationId

The `paypageRegistrationId` element is a required child of the `paypage` element, and specifies the Registration ID generated by `securepaypage.litle.com` (eProtect). It can also be used in a Register Token Request to obtain a token based on eProtect 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

4.287 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>
```

4.288 payPalNotes

The `payPalNotes` element is an optional child of multiple transaction types. You use this field to record additional information about the PayPal transaction.

Type = String; **Type** = String; **minLength** = N/A; **maxLength** = 255

Parent Elements:

[authReversal](#), [capture](#), [credit](#), [sale](#)

Attributes:

None

Child Elements:

None

4.289 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

4.290 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.

4.290.0.1 phone as a child of billToAddress and shipToAddress

The phone element defines the customer's phone number in both the billToAddress and shipToAddress elements.

NOTE: When submitting an eCheck Verification, the string can only contain numbers (0 through 9). Letters and special characters are not allowed.

Type = String; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[billToAddress](#), [shipFromPostalCode](#)

Attributes:

None

Child Elements:

None

4.290.0.2 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

4.291 physicalCheckCredit

The `physicalCheckCredit` element is the parent element for the transaction type that a PayFac uses to distribute funds to a third party who issues physical checks on the PayFac's behalf. (i.e., from the PayFac Settlement Account to the Third Party Account).

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#)

4.292 physicalCheckCreditResponse

The `physicalCheckCreditResponse` element is the parent element for information returned to you in response to a `physicalCheckCredit` transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.293 physicalCheckDebit

The `physicalCheckDebit` element is the parent element for the transaction type that a PayFac uses to move funds from a third party who issues physical checks on the PayFac's behalf. (i.e., from the Third Party Account to the PayFac Settlement Account).

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#)

4.294 physicalCheckDebitResponse

The `physicalCheckDebitResponse` element is the parent element for information returned to you in response to a `physicalCheckDebit` transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.295 pin

The `pin` element is an optional child of the several transaction types, as well as the `card` element. It only applies to transactions involving closed-loop Gift Cards and defines the pin number associated with the Gift Card.

Type = String; **minLength** = 4; **maxLength** = 12

Parent Elements:

[capture](#), [card](#), [credit](#), [virtualGiftCardResponse](#)

Attributes:

None

Child Elements:

None

4.296 planCode

The `planCode` element is the identifier of a defined recurring payment plan. You use it to specify the payment plan when submitting a recurring transaction to the Recurring Engine. For example, there could be a define plan called **Monthly** that instructs the Recurring Engine to bill the consumer the same amount every month for the number of months defined by the `numberOfPayments` element. This element is a required child of the `subscription` element.

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

Parent Elements:

[subscription](#), [updateSubscription](#), [createPlan](#), [updatePlan](#), [updatePlanResponse](#)

Attributes:

None

Child Elements:

None

4.297 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](#), [terminalId](#), [catLevel](#)

Example: pos Structure

```
<pos>
    <capability>Capabilty Enumeration</capability>
    <entryMode>Entry Mode Enumeration</entryMode>
    <cardholderId>Cardholder ID Enumeration</cardholderId>
    <terminalId>1234567890</terminalId>
    <catLevel>Capabilty of CAT Terminal</catLevel>
</pos>
```

NOTE:

For CAT (Cardholder Activated Terminal) transactions, the capability element must be set to magstripe, the cardholderId element must be set to nopin, and the catLevel element must be set to self service.

4.298 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:

[activateResponse](#), [activateReversalResponse](#), [authorizationResponse](#), [authReversalResponse](#), [captureResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [deactivateResponse](#), [deactivateReversalResponse](#), [depositReversalResponse](#), [echeckCreditResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [forceCaptureResponse](#), [loadResponse](#), [loadReversalResponse](#), [refundReversalResponse](#), [saleResponse](#), [unloadResponse](#), [unloadReversalResponse](#), [voidResponse](#)

Attributes:

None

Child Elements:

None

4.299 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 the system created the Account Updater acknowledgment file.

Type = Date; minLength = N/A; maxLength = 10

Parent Elements:

[accountUpdateFileRequestData](#)

Attributes:

None

Child Elements:

None

4.300 preapprovalNumber

The `preapprovalNumber` element is an optional child of the `billMeLaterRequest` element, which you use to specify the pre-approval number issued by PayPal Credit. 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.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

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

Parent Elements:

[billMeLaterRequest](#)

Attributes:

None

Child Elements:

None

4.301 preferredLanguage

The `preferredLanguage` element is an optional child of the `sepaDirectDebit` element. This defines the language in which the merchant prefers the mandate to appear. While the merchant could be able to select any language, the mandate may not be available in the selected language. If the selected language is not available, the mandate appears in English. If you do not include this element, the preferred language defaults to the language indicated by the country of the IBAN, unless it is not available, in which case the language defaults to English.

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.

All Country Codes are 2-character except for the United States, which accepts both US and USA.

Parent Elements:

[sepaDirectDebit](#), [ideal](#), [giropay](#)

Attributes:

None

Child Elements:

None

4.302 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

4.303 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

4.304 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>
```

4.305 processingType

The processingType element is an optional child of several transaction types. You use this element to define a Visa transaction is intended to fund a host-based prepaid product, a brokerage account, or an escrow account.

Type = String (Enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

authorization, captureGivenAuth, forceCapture, sale

Attributes:

None

Child Elements:

None

Enumerations:

Enum	Description
accountFunding	Use this value to define a Visa transaction is intended to fund a host-based prepaid product, a brokerage account, or an escrow account.
initialRecurring	Use this value to specify the first in a series of recurring payments, when the transaction involves the use of a Network Token, such as an Apple Pay or Android Pay transaction.
initialInstallment	Use this value to specify the first in a series of installment payments, when the transaction involves the use of a Network Token, such as an Apple Pay or Android Pay transaction.

4.306 productCode

The `productCode` element is an optional child of the `lineItemData` element, which specifies the product code of the purchased item. This value is a merchant defined description code of the product/service. This could be an inventory number, UPC, catalog number, or some other value that the merchant uses to define the specific product. 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

4.307 publicKeyHash

The `publicKeyHash` element is a required child of the `header` element and provides the BASE64 Encoded string that is a hash of the merchant's certificate public key bytes associated with the Apple Pay transaction.

Type = Base 64 Encoded String; **minLength** = N/A; **maxLength** = 200

Parent Elements:

[header](#)

Attributes:

None

Child Elements:

None

4.308 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.

NOTE: If you accidentally omit the `quantity` element, our system will submit the transaction using a default value of 1.

Type = Decimal; **minInclusive** = 0; **totalDigits** = 12

Parent Elements:

[lineItemData](#)

Attributes:

None

Child Elements:

None

4.309 queryTransaction

The `queryTransaction` element is the parent element for Query transactions. You use this transaction type to determine the status of a previously submitted transaction. You can submit this element only as an Online transaction.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: `origId`, `origActionType`

Optional: `origLittleTxnId`, `origOrderId`, `origAccountNumber`

4.310 queryTransactionResponse

The `queryTransactionResponse` element is the parent element for the response to `queryTransaction` requests.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

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

Child Elements:

All Required: `response`, `responseTime`, `message`, `matchCount`, `results_Max10`

4.311 queryTransactionUnavailableResponse

The `queryTransactionUnavailableResponse` element is an optional child of the `results_Max10` element. If the query results is a response code of 152 - Transaction found, but response not yet available, the `results_Max10` element will contain at least one `queryTransactionUnavailableResponse` child (see example *results_Max10 Element for 152 Response Code*).

Parent Elements:

[results_Max10](#)

Attributes:

None

Child Elements:

All Required: [littleTxnId](#), [response](#), [message](#)

4.312 recurringRequest

The `recurringRequest` element is the parent of several child element that define the number of payments and plan type of recurring transaction to be handled by the Recurring Engine. It is an optional child of the Authorization and Sale transactions.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

[subscription](#)

Example: recurringRequest Structure

```
<recurringRequest>
    <subscription>
        <planCode>Plan Reference Code</planCode>
        <numberOfPayments>1 to 99</numberOfPayments>
        <startDate>Start Date of Recurring Cycle</startDate>
        <amount>Amount of Recurring Payment</amount>
    </subscription>
</recurringRequest>
```

4.313 recurringResponse

The recurringResponse element is the parent element for the subscriptionId, responseCode, responseMessage, and recurringTxnId elements associated with a requested recurring payment. The system returns this element only when the sale transaction includes a recurringRequest element.

Parent Elements:

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

Attributes:

None

Child Elements:

[subscriptionId](#), [responseCode](#), [responseMessage](#), [recurringTxnId](#)

Example: recurringResponse Structure

```
<recurringResponse>
    <subscriptionId>1234567890</subscriptionId>
    <responseCode>Response Code</responseCode>
    <responseMessage>Response Message</responseMessage>
</recurringResponse>
```

4.314 recurringTxnId

The `recurringTxnId` element is an optional child of the `recurringResponse` element used to identify the record of recurring, scheduled transactions.

Type = Long; **minLength** = N/A; **maxLength** = 19

Parent Elements:

[recurringResponse](#), [littleInternalRecurringRequest](#)

NOTE: Although the element is an optional child of the `recurringResponse` element, it will never be returned in the response to a merchant initiated transaction.

Attributes:

None

Child Elements:

None

4.315 recycleAdvice

NOTE: The recycling Advice feature is no longer supported.

The recyclingAdvice element contains 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>2016-11-15T12:00:00</nextRecycleTime>
</recycleAdvice>
```

Example: recycleAdvice Structure - with end message

```
<recycleAdvice>
    <recycleAdviceEnd>End of Advice</recycleAdviceEnd>
</recycleAdvice>
```

4.316 recycleAdviceEnd

NOTE: The recycling Advice feature is no longer supported.

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

4.317 recycleBy

The `recycleBy` element is an optional child of the `recyclingRequest` element and determines the use of the Recycling Engine. The default setting is `Little`, so omitting this element is the same as submitting a value of `Little`.

NOTE:

Also, if your Merchant Profile includes a preset percentage split of transactions between merchant and Vantiv controlled, then settings of Merchant and Little are ignored; you can still use 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 Recycling Engine controls the recycling of the transaction. For A/B comparison testing, transactions using this setting will be counted as Vantiv 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 Vantiv controlled.

4.318 recycleEngineActive

The `recycleEngineActive` element is an optional child of the `recycling` element that indicates whether or not the engine is recycling the declined transaction. This element is returned only if you are using the Recycling Engine.

Type = Boolean; **Valid values** = true or false

Parent Elements:

[recycling](#)

Attributes:

None

Child Elements:

None

4.319 recycleId

The `recycleId` element is an optional child of the `recyclingRequest` element. Merchants can use this identifier as part of the transaction signature used to track the recycling of a transaction. This element is an alternative to using the `orderId` element as part of the transaction signature.

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

Parent Elements:

[recyclingRequest](#)

Attributes:

None

Child Elements:

None

4.320 recycling

The `recycling` element has two uses in LittleXML depending upon the parent. When used as a child of either the `authorizationResponse` or `saleResponse` elements, the `recycling` element contains a child element that specifies whether or not the engine is recycling the declined transaction.

When used as a child of the `voidResponse`, the `recycling` element contains a child providing the Transaction Id of the Credit transaction issued. This occurs only if a Void transaction is used to halt the recycling of a transaction by the recycling engine and the transaction has already been approved and captured (see [Using Void to Halt Recycling Engine](#) on page 85).

4.320.1 recycling Element as a Child of `authorizationResponse` or `saleResponse`

The `recycling` element contains child elements indicating that the Recycling Engine is active.

Parent Elements:

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

Attributes:

None

Child Elements:

[recycleAdvice](#), [recycleEngineActive](#)

NOTE: The recycling Advice feature is no longer supported.

Example: recycling Structure - with engine active flag

```
<recycling>
    <recycleEngineActive>true or false</recycleEngineActive>
</recycling>
```

4.320.2 recycling Element as a Child of `voidResponse`

The `recycling` element is an optional child of the `voidResponse` element and contains a child providing the Transaction Id of the Credit transaction issued. This element is present in the Void response only under the following circumstances (see [Using Void to Halt Recycling Engine](#) on page 85):

- You submitted a Void transaction to halt the recycling of a declined Sale transaction by the Recovery/Recycling Engine.
- The Sale transaction has already been approved and captured.
- Your Recovery/Recycling Engine configuration enables automatic refunds.
- The system has successfully submitted a Credit transaction on your behalf.

Parent Elements:[voidResponse](#)**Attributes:**

None

Child Elements:[creditLittleTxnId](#)**Example: recycling Structure - child of voidResponse**

```
<recycling>
    <creditLittleTxnId>1234567890123456789</creditLittleTxnId>
</recycling>
```

4.321 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. It also contains an optional element for an identifier assigned by the merchant to track the recycling of the transaction. This element only applies to merchants using the Recycling Engine.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

[recycleBy](#), [recycleId](#)

Example: recyclingRequest Structure

```
<recyclingRequest>
    <recycleBy>Merchant or Little or None</recycleBy>
    <recycleId>abcdef1234567890</recycleId>
</recyclingRequest>
```

4.322 redirectToken

The `redirectToken` element is an optional child of the `sepaDirectDebitResponse`, `idealResponse`, and `giropayResponse` elements. It allows you to verify that the consumer accepted the Mandate by comparing it to the `token_value` parameter returned in the URL upon redirect from the Mandate site.

Type = String; **minLength** = N/A; **maxLength** = 128

Parent Elements:

[sepaDirectDebitResponse](#), [idealResponse](#), [giropayResponse](#)

Attributes:

None

Child Elements:

None

4.323 redirectUrl

The `redirectUrl` element is an optional child of the `sepaDirectDebit` element and defines the URL that hosts the mandate, when Vantiv supplies the mandate. If you supply the mandate (`<mandateProvider>Merchant</mandateProvider>`), this element will not appear in the response.

Type = String; **minLength** = N/A; **maxLength** = 256

Parent Elements:

[sepaDirectDebitResponse](#), [idealResponse](#), [giropayResponse](#)

Attributes:

None

Child Elements:

None

4.324 refCode

The `refCode` element is an optional child of the `giftCardResponse` element, where it specifies the authorization code of the gift card transaction.

Type = String; **minLength** = N/A; **maxLength** = 6

Parent Elements:

[giftCardResponse](#)

Attributes:

None

Child Elements:

None

4.325 refundReversal

The `refundReversal` element is the parent element for a Gift Card specific transaction type that reverses the a refund associated with a Gift Card.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (Required)

[card](#), [originalRefCode](#), [originalAmount](#), [originalTxnTime](#), [originalSystemTraceId](#), [originalSequenceNumber](#)

Child Elements: (Optional)

[littleTxnId](#)

4.326 refundReversalResponse

The `refundReversalResponse` element is the parent element for information returned to you in response to an `refundReversal` transaction.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#), [giftCardResponse](#)

Optional: [postDate](#)

4.327 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
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for information. minLength = 1 maxLength = 25

Child Elements:

Required: either [accountNumber](#), [echeckForToken](#), [mpos](#), [paypageRegistrationId](#), or [applepay](#)

Optional: [orderId](#), [cardValidationNum](#)

NOTE: The use of the `cardValidationNum` element in the `registerTokenRequest` only applies when you submit an `accountNumber` element.

4.328 registerTokenResponse

The `registerTokenResponse` element is the parent element for the 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
<code>id</code>	String	Yes	The response returns the same value submitted in the <code>registerTokenRequest</code> transaction. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	The response returns the same value submitted in the <code>registerTokenRequest</code> transaction. minLength = N/A maxLength = 50
<code>reportGroup</code>	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](#), [littleToken](#), [bin](#), [type](#), [applepayResponse](#), [androidpayResponse](#)

4.329 reloadable

The `reloadable` element is an optional child of the `fundingSource` element and defines whether the prepaid card is reloadable.

NOTE: This element is never returned for American Express card transaction.

Type = String (Enum); **Enumerations** = YES, NO, or UNKNOWN

Parent Elements:

`fundingSource`

Attributes:

None

Child Elements:

None

4.330 reserveCredit

The `reserveCredit` element is the parent element for the transaction type that a PayFac uses to move funds from the PayFac Settlement Account to the PayFac Reserve Account.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#)

4.331 reserveCreditResponse

The `reserveCreditResponse` element is the parent element for information returned to you in response to a `reserveCredit` transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.332 reserveDebit

The `reserveDebit` element is the parent element for the transaction type that a PayFac uses to move funds from the PayFac Reserve Account to the PayFac Settlement Account.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#)

4.333 reserveDebitResponse

The `reserveDebitResponse` element is the parent element for information returned to you in response to a `reserveDebit` transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.334 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 PayPal Credit transactions.

NOTE:

As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = String (Enum); **Enumerations** = Own, Rent, or Other

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.335 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 response codes and associated messages, please refer to [Payment Transaction Response Codes](#) on page 812.

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

Parent Elements:

[activateResponse](#), [activateReversalResponse](#), [authorizationResponse](#), [authReversalResponse](#), [captureResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [deactivateResponse](#), [deactivateReversalResponse](#), [depositReversalResponse](#), [echeckCreditResponse](#), [echeckPreNoteCreditResponse](#), [echeckPreNoteSaleResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [echeckVoidResponse](#), [forceCaptureResponse](#), [fraudCheckResponse](#), [loadResponse](#), [loadReversalResponse](#), [queryTransactionResponse](#), [queryTransactionUnavailableResponse](#), [registerTokenResponse](#), [refundReversalResponse](#), [saleResponse](#), [voidResponse](#), [cancelSubscriptionResponse](#), [unloadResponse](#), [updatePlanResponse](#), [updateSubscriptionResponse](#), [unloadReversalResponse](#), [payFacCreditResponse](#), [payFacDebitResponse](#), [physicalCheckCreditResponse](#), [physicalCheckDebitResponse](#), [submerchantCreditResponse](#), [reserveCreditResponse](#), [reserveDebitResponse](#), [submerchantDebitResponse](#), [vendorCreditResponse](#), [vendorDebitResponse](#)

Attributes:

None

Child Elements:

None

4.336 responseCode

The **responseCode** element contains a three digit numeric code which along with the **responseMessage** element specifies either acceptance by the Recurring Engine or the reason the recurring Engine was unable to schedule subsequent payments.

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

Parent Elements:

[recurringResponse](#)

Attributes:

None

Child Elements:

None

4.337 responseMessage

The **responseMessage** element contains a brief definition of the **responseCode** returned for the recurring transaction.

Type = String; **minLength** = N/A; **maxLength** = 512

Parent Elements:

[recurringResponse](#)

Attributes:

None

Child Elements:

None

4.338 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:

[activateResponse](#), [activateReversalResponse](#), [authorizationResponse](#), [authReversalResponse](#), [captureResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [deactivateResponse](#), [deactivateReversalResponse](#), [depositReversalResponse](#), [echeckCreditResponse](#), [echeckPreNoteCreditResponse](#), [echeckPreNoteSaleResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [echeckVoidResponse](#), [forceCaptureResponse](#), [fraudCheckResponse](#), [loadResponse](#), [loadReversalResponse](#), [queryTransactionResponse](#), [registerTokenResponse](#), [refundReversalResponse](#), [saleResponse](#), [voidResponse](#), [cancelSubscriptionResponse](#), [unloadResponse](#), [unloadReversalResponse](#), [updatePlanResponse](#), [updateSubscriptionResponse](#), [payFacCreditResponse](#), [payFacDebitResponse](#), [physicalCheckCreditResponse](#), [physicalCheckDebitResponse](#), [reserveCreditResponse](#), [reserveDebitResponse](#), [submerchantCreditResponse](#), [submerchantDebitResponse](#), [vendorCreditResponse](#), [vendorDebitResponse](#)

Attributes:

None

Child Elements:

None

4.339 results_Max10

The `results_Max10` element is a required child of the `queryTransactionResponse`. Any original transaction responses that match the criteria submitted in the `queryTransaction` appear as children of this element in the response. The system can return a maximum of ten responses as children of the `results_Max10` element. The value for the `matchCount` element reflects the number of found transactions.

If the system does not find any transactions matching the query criteria, the `results_Max10` element will be empty.

If the query results is a response code of 152 - Transaction found, but response not yet available, the `results_Max10` element will contain at least one `queryTransactionUnavailableResponse` child and may contain other found responses.

Parent Elements:

`queryTransactionResponse`

Attributes:

None

Child Elements:

All Optional: `activateResponse`, `activateReversalResponse`, `authorizationResponse`, `captureResponse`, `creditResponse`, `deactivateResponse`, `depositReversalResponse`, `echeckCreditResponse`, `echeckSalesResponse`, `loadResponse`, `loadReversalResponse`, `refundReversalResponse`, `saleResponse`, `unloadResponse`, `unloadReversalResponse`, `voidResponse`, `queryTransactionUnavailableResponse`

Example: results_Max10 Element with One Found Transaction

```
<results_max10>
  <authorizationResponse id="GiftCardAuth" reportGroup="Mer5PM1" customerId="1">
    <littleTxnId>82827170811986124</littleTxnId>
    <orderId>150330_GCAuth</orderId>
    <response>000</response>
    <responseTime>2015-04-06T16:40:04</responseTime>
    <postDate>2015-04-06</postDate>
    <message>Approved</message>
    <authCode>111115</authCode>
    <fraudResult>
      <avResult>30</avResult>
```

```
<cardValidationResult>M</cardValidationResult>
</fraudResult>
<giftCardResponse>
  <availableBalance>125</availableBalance>
</giftCardResponse>
</authorizationResponse>
</results_max10>
```

Example: results_Max10 Element with Multiple Found Transactions

```
<results_max10>
  <authorizationResponse id="DupeId" reportGroup="Mer5PM1">
    <littleTxnId>82827170811986215</littleTxnId>
    <orderId>150331_DupeAuth2</orderId>
    <response>000</response>
    <responseTime>2015-04-06T16:40:12</responseTime>
    <postDate>2015-04-06</postDate>
    <message>Approved</message>
    <authCode>055858</authCode>
    <fraudResult>
      <avsResult>32</avsResult>
      <cardValidationResult>M</cardValidationResult>
    </fraudResult>
  </authorizationResponse>
  <authorizationResponse id="DupeId" reportGroup="Mer5PM1">
    <littleTxnId>82827170811986207</littleTxnId>
    <orderId>150331_DupeAuth1</orderId>
    <response>000</response>
    <responseTime>2015-04-06T16:40:11</responseTime>
    <postDate>2015-04-06</postDate>
    <message>Approved</message>
    <authCode>111111</authCode>
    <fraudResult>
      <avsResult>00</avsResult>
      <cardValidationResult>M</cardValidationResult>
    </fraudResult>
  </authorizationResponse>
</results_max10>
```

```
</authorizationResponse>  
</results_max10>
```

Example: results_Max10 Element for 152 Response Code

```
<results_max10>  
  <queryTransactionUnavailableResponse>  
    <littleTxnId>82827170811986124</littleTxnId>  
    <response>152</response>  
    <message>Original transaction found but response not yet available</message>  
  </queryTransactionUnavailableResponse>  
</results_max10>
```

4.340 RFRequest

The RFRequest 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 RFResponse 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: RFRequest Structure - Batch

```
<RFRequest>
    <littleSessionId>Session ID</littleSessionId>
</RFRequest>
```

Example: RFRequest Structure - Account Updater

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

4.341 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

4.342 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** = 8; **maxLength** = 9

Parent Elements:

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

Attributes:

None

Child Elements:

None

NOTE: If you submit an invalid routing number, we return the XML Response
Code 900 - Invalid Bank Routing Number.

4.343 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

4.344 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:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [orderId](#), [amount](#), [orderSource](#), (choice of) [card](#), [paypal](#), [paypage](#), [mpos](#), [token](#), [applepay](#), [ideal](#), [sepaDirectDebit](#), [giropay](#), or [sofort](#)

NOTE: The `cardholderAuthentication` child element is required only for 3-D Secure transactions and for BML ecommerce transactions.
The `fraudCheck` element has been deprecated; use the `cardholderAuthentication` element instead.

Optional: [littleTxnId](#), [customerInfo](#), [billToAddress](#), [shipToAddress](#), [billMeLaterRequest](#), [cardholderAuthentication](#), [customBilling](#), [taxType](#), [enhancedData](#), [processingInstructions](#), [pos](#), [payPalNotes](#), [payPalOrderComplete](#), [amexAggregatorData](#), [allowPartialAuth](#), [healthcareIIS](#), [merchantData](#), [recyclingRequest](#), [fraudFilterOverride](#), [secondaryAmount](#), [surchargeAmount](#), [recurringRequest](#), [littleInternalRecurringRequest](#), [debtRepayment](#), [advancedFraudChecks](#), [wallet](#), [processingType](#), [origOrderId](#), [password](#), [originalNetworkTransactionId](#), [originalTransactionAmount](#)

4.345 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
<code>id</code>	String	Yes	The response returns the same value submitted in the Sale transaction. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	The response returns the same value submitted in the Sale transaction. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	The response returns the same value submitted in the Sale transaction. minLength = 1 maxLength = 25

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](#), [recurringResponse](#), [giftCardResponse](#), [applepayResponse](#), [cardSuffix](#), [androidpayResponse](#), [sepaDirectDebitResponse](#), [networkTransactionId](#), [idealResponse](#), [giropayResponse](#), [sofortResponse](#)

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 Relationship Manager for additional information.

The `authorizationResponseSubCode` element is not used at this time.

4.346 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

4.347 secondaryAmount

The secondaryAmount element defines the principal portion of the total amount when a convenience fee applied to the transaction by the merchant. for example, if the total charge is \$105, with the principal amount being \$100 and the convenience fee being \$5, you must use \$100 as the value for the secondaryAmount element. Supply the value in cents without a decimal point. For example, a value of 400 signifies \$4.00.

Type = Integer; **totalDigits** = 12

Parent Elements:

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

Attributes:

None

Child Elements:

None

4.348 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

4.349 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

4.350 sepaDirectDebit

The `sepaDirectDebit` element is a child of the `sale` transaction that, through its child elements, defines information needed to process a SEPA Direct Debit transaction. At this time, you can use the SEPA Direct Debit method of payment in Online transactions only.

Parent Elements:

[sale](#)

Attributes:

None

Child Elements:

Required: [iban](#), [mandateProvider](#), [sequenceType](#)

Optional: [mandateReference](#), [mandateURL](#), [mandateSignatureDate](#), [preferredLanguage](#)

Example: sepaDirectDebit Structure

```
<sepaDirectDebit>
    <mandateProvider>Merchant or Vantiv</mandateProvider>
    <sequenceType>One Time or Type of Recurring</sequenceType>
    <mandateReference>Ref to First Payment of Recurring Stream</mandateReference>
    <mandateUrl>URL of Merchant Hosted Mandate</mandateUrl>
    <mandateSignatureDate>2016-06-12</mandateSignatureDate>
    <iban>Consumer Account Number</iban>
    <preferredLanguage>Country Code of Language</preferredLanguage>
</sepaDirectDebit>
```

4.351 sepaDirectDebitResponse

The `sepaDirectDebitResponse` element is a child of the `saleResponse` transaction when the method of payment in the request is `sepaDirectDebit`. It contains child elements that you should store for future use/reference.

Parent Elements:

[saleResponse](#)

Attributes:

None

Child Elements (all Optional):

[mandateReference](#), [redirectUrl](#), [redirectToken](#)

Example: sepaDirectDebitResponse Structure

```
<sepaDirectDebitResponse>
    <redirectUrl>URL of Vantiv Supplied Mandate</redirectUrl>
    <redirectToken>Map Mandate to littleTxnId</redirectToken>
    <mandateReference>Ref Number for Subsequent Payments</mandateReference>
</sepaDirectDebit>
```

4.352 sequenceNumber

The sequenceNumber element is an optional child of the giftCardResponse element, which specifies a Vantiv generated sequence number associated with the transaction in our systems. You should retain this value for possible future use in gift card reversal transactions.

Type = String; **totalDigits** = 6; **Allowed Characters:** 0 - 9

Parent Elements:

[giftCardResponse](#)

Attributes:

None

Child Elements:

None

4.353 sequenceType

The sequenceType element is a required child of the sepaDirectDebit element and defines the purchase in terms of a one-time buy or a member of a recurring stream of debits. If the value of this element is either **SubsequentRecurring**, or **FinalRecurring**, you must include the mandateReference element in the request, specifying the value returned in the initial sepaDirectDebitResponse.

Type = String (Enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[sepaDirectDebit](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
OneTime	The purchase is a one-time buy (i.e., not part of a recurring stream).
FirstRecurring	This purchase is the initial buy of a recurring stream.
SubsequentRecurring	This purchase is part of a recurring stream of purchases, but not the first or last in the stream.
FinalRecurring	This purchase is the final buy of a recurring stream.

4.354 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 `<shipFromPostalCode>` element as 20 characters, in practice you should never exceed 10 characters in your submissions.

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.355 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.

Type = Integer; **totalDigits** = 8

Parent Elements:

[enhancedData](#)

Attributes:

None

Child Elements:

None

4.356 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](#), [fraudCheck](#), [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>
```

4.357 signature

The `signature` element is a required child of the `applepay` element. It is the BASE64 encoded string signature of the payment and header data from the PKPaymentToken.

Type = String; **minLength** = N/A; **maxLength** = 10000

Parent Elements:

[applepay](#)

Attributes:

None

Child Elements:

None

4.358 sofort

The `sofort` element is a child of the `sale` transaction that, through its child elements, defines information needed to process an SOFORT (Real-time Bank Transfer) transaction. At this time, you can use the iDeal method of payment in Online transactions only.

Parent Elements:

[sale](#)

Attributes:

None

Child Elements:

Optional: [preferredLanguage](#)

Example: sofort Structure

```
<sofort>
    <preferredLanguage>Country Code of Language</preferredLanguage>
</sofort>
```

4.359 sofortResponse

The `sofortResponse` element is a child of the `saleResponse` transaction when the method of payment in the request is `sofort`. It contains child elements that you should store for future use/reference.

Parent Elements:

[saleResponse](#)

Attributes:

None

Child Elements (all Optional):

[paymentPurpose](#), [redirectUrl](#), [redirectToken](#)

Example: sofortResponse Structure

```
<sofortResponse>
    <redirectUrl>URL of Vantiv Supplied Mandate</redirectUrl>
    <redirectToken>Map Mandate to littleTxnId</redirectToken>
    <paymentPurpose>Reference Number + Billing Descriptor</paymentPurpose>
</sofortResponse>
```

4.360 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 PayPal Credit transactions.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

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

4.361 startDate

The `startDate` element is a optional child of the `subscription` element, which specifies the date the recurring billing should begin. It is also an optional child of both the `createAddOn` and `createDiscount` element, where it specifies either the starting date of the Add On charge or the starting date of the discount.

Type = Date; **Format** = YYYY-MM-DD

Parent Elements:

[subscription](#), [createAddOn](#), [createDiscount](#), [updateAddOn](#), [updateDiscount](#)

Attributes:

None

Child Elements:

None

4.362 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

4.363 submerchantCredit

The submerchantCredit element is the parent element for the transaction type that a PayFac uses to move funds from the PayFac Settlement Account to the Sub-merchant Account.

NOTE: For additional information about PayFac Instruction-Based Dynamic Payout and the use of this transaction type, please refer to the *PayFac Instruction-Based Dynamic Payout* document.

Parent Elements:

batchRequest

Attributes:

Attribute Name	Type	Required?	Description
id	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
reportGroup	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [accountInfo](#), [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#), [submerchantName](#), [customIdentifier](#)

4.364 submerchantCreditResponse

The submerchantCreditResponse element is the parent element for information returned to you in response to a submerchantCredit transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.365 submerchantDebit

The `submerchantDebit` element is the parent element for the transaction type that a PayFac uses to move funds from the Sub-merchant Account to the PayFac Settlement Account.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [accountInfo](#), [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#), [submerchantName](#), [customIdentifier](#)

4.366 submerchantDebitResponse

The submerchantDebitResponse element is the parent element for information returned to you in response to a submerchantDebit transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.367 submerchantName

The submerchantName element is a required child of the submerchantCredit element and specifies the name of the Sub-merchant.

Type = String; **minLength** = 1; **maxLength** = 256

Parent Elements:

[submerchantCredit](#), [submerchantDebit](#)

Attributes:

None

Child Elements:

None

4.368 subscription

The subscription element is a required child of the recurringRequest element and the parent of several child element that define information about the recurring transaction stream to be handled by the Recurring Engine.

Parent Elements:

[recurringRequest](#)

Attributes:

None

Child Elements (required):

[planCode](#)

Child Elements (optional):

[numberOfPayments](#), [startDate](#), [amount](#), [createDiscount](#), [createAddOn](#)

NOTE: If you include the `numberOfPayments` child element, the value submitted overrides the default value defined in the Plan.

Example: subscription Structure

```
<subscription>
    <planCode>Plan Reference Code</planCode>
    <numberOfPayments>1 to 99</numberOfPayments>
    <startDate>Start Date of recurring Cycle</startDate>
    <amount>Amount of Recurring Payment</amount>
    <createDiscount>
        <discountCode>Discount Reference Code</addOnCode>
        <name>Name of Discount</name>
        <amount>Amount of Discount</amount>
        <startDate>Start Date of Discount</startDate>
        <endDate>End Date of Discount</endDate>
    </createDiscount>
    <createAddOn>
        <addOnCode>Add On Reference Code</addOnCode>
```

```
<name>Name of Add On</name>
<amount>Amount of Add On</amount>
<startDate>Start Date of Add On Charge</startDate>
<endDate>End Date of Add On Charge</endDate>
</createAddOn>
</subscription>
```

4.369 subscriptionId

The `subscriptionId` element is a required child of the `recurringResponse` element and defines the assigned identifier for the sequence of recurring billing transactions. You also use this element in the `updateSubscription` and `cancelSubscription` transactions to identify the subscription for changes/cancellation.

Type = Long; **minLength** = N/A; **maxLength** = 19

Parent Elements:

`recurringResponse`, `littleInternalRecurringRequest`, `cancelSubscription`, `updateSubscription`, `updateSubscriptionResponse`, `cancelSubscriptionResponse`

Attributes:

None

Child Elements:

None

4.370 surchargeAmount

The `surchargeAmount` element defines the amount of the surcharge applied to the transaction by the merchant. Supply the value in cents without a decimal point. For example, a value of 400 signifies \$4.00.

Type = Integer; totalDigits = 12

NOTE: Use of the `surchargeAmount` element applies to Visa or MasterCard credit card payments only. Also, you are required to notify the card networks and us of your intent to applying surcharges at least 30 days prior to implementing the surcharges. Please consult your Relationship Manager if you have additional questions.

Parent Elements:

[authorization](#), [authReversal](#), [capture](#), [credit](#), [captureGivenAuth](#), [forceCapture](#), [sale](#)

Attributes:

None

Child Elements:

None

4.371 systemTraceId

The `systemTraceId` element is an optional child of the `giftCardResponse` element, which specifies a Vantiv generated identifier associated with the transaction in our systems. You should retain this value for possible future use in gift card reversal transactions.

Type = Integer; **totalDigits** = 6

Parent Elements:

[giftCardResponse](#)

Attributes:

None

Child Elements:

None

4.372 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 `taxAmount` as a child of `lineItemData` along with `detailTax`, the `lineItemData` `taxAmount` should be the sum of the `taxAmount` children from `detailTax` children.

Attributes:

None

Child Elements:

None

4.373 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

4.374 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

4.375 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

4.376 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

4.377 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

4.378 terminalId

The `terminalId` element is an optional child of the `pos` element and defines the identifier of the terminal used at the point of sale.

NOTE: The `terminalId` element is required for MasterCard POS transactions.

Type = String; **minLength** = N/A; **maxLength** = 8 (No special characters allowed.)

Parent Elements:

[pos](#)

Attributes:

None

Child Elements:

None

4.379 termsAndConditions

The `termsAndConditions` element is an optional child of the `billMeLaterRequest` element and defines the specific lending terms of the PayPal Credit account.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = Integer; **totalDigits** = 5

Parent Elements:

[billMeLaterRequest](#)

Attributes:

None

Child Elements:

None

4.380 threatMetrixSessionId

The `threatMetrixSessionId` element is an optional child of the `advancedFraudChecks` element.

Type = String; **Allowed Characters** = a-z, A-Z, 0-9, -, _ ; **minLength** = 1; **maxLength** = 128

NOTE: While generated by you at the time the consumer accesses your page, each `threatMetrixSessionId` must include a 5-character prefix, supplied by your Implementation Consultant, followed by a dash (" - "). The remainder of the Id must be unique for each instance of the customer accessing your page.

Parent Elements:

[advancedFraudChecks](#)

Attributes:

None

Child Elements:

None

4.381 token

The `token` element has two uses depending upon whether the element concerns a Vantiv generated token (for tokenized merchants) or a PayPal generated token.

4.381.1 token (Vantiv 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](#), [updateSubscription](#)

Attributes:

None

Child Elements:

Required: [littleToken](#)

Optional: [expDate](#), [cardValidationNum](#), [type](#)

4.381.2 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

4.382 tokenMessage

The `tokenMessage` element provides a short, human-readable explanation of the `tokenResponseCode` (see [Table 4-3](#)).

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

Parent Elements:

[tokenResponse](#)

Attributes:

None

Child Elements:

None

4.383 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 or eCheck account information in the transaction request.

Parent Elements:

[authorizationResponse](#), [captureGivenAuthResponse](#), [creditResponse](#), [echeckCreditResponse](#), [echeckRedepositResponse](#), [echeckSalesResponse](#), [echeckVerificationResponse](#), [forceCaptureResponse](#), [saleResponse](#), [updateSubscriptionResponse](#)

Attributes:

None

Child Elements:

Required: `tokenResponseCode`, `tokenMessage`

Optional: `littleToken`, `type`, `bin`, `eCheckAccountSuffix`

Example: `tokenResponse` Structure

```
<tokenResponse>
    <littleToken>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>
```

4.384 tokenResponseCode

The `tokenResponseCode` element provides a 3-digit code (see [Table 4-3](#)) 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 4-3 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

4.385 totalHealthcareAmount

The `totalHealthcareAmount` element is a required child of the `healthcareAmounts` element and defines the total amount of healthcare related purchases. This value must be the greater than or equal to 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

4.386 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

4.387 track1Status

The `track1Status` element is a required child of the `mpos` element. This element indicates whether the device read track 1 from the magnetic stripe. A value of 0 indicates a successful read, while a value of 1 indicates a failure.

Type = Integer; **minInclusive** = 0; **maxInclusive** = 1028

Parent Elements:

[mpos](#)

Attributes:

None

Child Elements:

None

4.388 track2Status

The `track2Status` element is a required child of the `mpos` element. This element indicates whether the device read track 2 from the magnetic stripe. A value of 0 indicates a successful read, while a value of 1 indicates a failure.

Type = Integer; **minInclusive** = 0; **maxInclusive** = 1028

Parent Elements:

[mpos](#)

Attributes:

None

Child Elements:

None

4.389 transactionAmount

The `transactionAmount` element is an optional child of the `applepayResponse` element and specifies the amount of the transaction. The decimal is implied. Example: 500 = \$5.00.

Type = Integer; **totalDigits** = 12

Parent Elements:

[applepayResponse](#)

Attributes:

None

Child Elements:

None

4.390 transactionId

The `transactionId` element is used in two locations: in PayPal transactions, as a child of the `paypal` element and in Apple Pay transactions as a child of the `header` element.

4.390.1 transactionId as a Child of the paypal element

The `transactionId` element is a required child of the `paypal` element, specifying the transaction Id returned from PayPal.

NOTE: The value of the `<transactionId>` element must match the `TRANSACTIONID` value returned by the `DoExpressCheckoutPayment` call operation to PayPal.

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

Parent Elements:

[paypal](#)

Attributes:

None

Child Elements:

None

4.390.2 transactionId as a Child of the header element

The `transctionId` element is a required child of the `header` element and provides the hexadecimal transaction identifier generated on the device for an Apple Pay transaction.

Type = Hex Encoded String; **minLength** = N/A; **maxLength** = 250

Parent Elements:

[header](#)

Attributes:

None

Child Elements:

None

4.391 trialIntervalType

The `trialIntervalType` element is an optional child of the `createPlan` element and defines the interval period of a trial associated with the Plan. The overall length of a trial period is defined by the `trialIntervalType` combined with the `trialNumberOfIntervals` element.

Type = String (enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[createPlan](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
MONTH	The trial interval one month.
DAY	The trial interval one day.

4.392 trialNumberOfIntervals

The `trialNumberOfIntervals` element is an optional child of the `createPlan` element and defines the number of trial intervals (`trialIntervalType`) associated with the Plan. The overall length of a trial period is defined by the `trialIntervalType` combined with the `trialNumberOfIntervals` element.

Type = Integer; **minLength** = 1; **maxLength** = 99

Parent Elements:

[createPlan](#)

Attributes:

None

Child Elements:

None

4.393 triggeredRule

The `triggeredRule` element is an optional child of the `advancedFraudResult` element. It can appear multiple times in the response, once for each triggered rule from the ThreatMetrix policy. A triggered rule is one where the threshold is exceeded.

Type = String; **minLength** = N/A; **maxLength** = 64

Parent Elements:

[advancedFraudResults](#)

Attributes:

None

Child Elements:

None

4.394 txnTime

The `txnTime` element is an optional child of the `giftCardResponse` element, which specifies the date and time the transaction was processed by Vantiv. You should retain this value for possible future use in other gift card transactions, such as `giftCardCapture` and most reversal transactions. The format of the element is YYYY-MM-DDTHH:MM:SS. For example, 2016-11-21T11:00:00.

Type = `dateTime`; **minLength** = N/A; **maxLength** = 20

Parent Elements:

[giftCardResponse](#)

Attributes:

None

Child Elements:

None

4.395 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, 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.

4.395.1 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, 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`, `token`

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
MC	MasterCard
VI	Visa
AX	American Express
DC	Diner's Club (see Table B-1)
DI	Discover
PP	PayPal
JC	JCB (Japanese Credit Bureau)
BL	PayPal Credit
EC	eCheck

Enumeration	Description
GC	Gift Card
"" (empty)	Card type unknown or undefined

4.395.2 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 Insights features (see [Customer Insight Features](#) on page 25.)

Please consult your Relationship Manager for additional information.

4.396 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

4.397 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

4.398 unload

The `unload` element is the parent element for the transaction type that removes funds from a Gift Card.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (all Required)

[amount](#), [orderSource](#), [card](#)

4.399 unloadResponse

The `unloadResponse` element is the parent element for information returned to you in response to an `unload` 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
<code>id</code>	String	Yes	The response returns the same value submitted in the Unload transaction. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	The response returns the same value submitted in the Unload transaction. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	The response returns the same value submitted in the Unload transaction. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [fraudResult](#), [giftCardResponse](#)

4.400 unloadReversal

The `unloadReversal` element is the parent element for the transaction type that reverses the unloading of a Gift Card.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements: (Required)

[card](#), [originalRefCode](#), [originalAmount](#), [originalTxnTime](#), [originalSystemTraceId](#), [originalSequenceNumber](#)

Child Elements: (Optional)

[littleTxnId](#)

4.401 unloadReversalResponse

The `unloadReversalResponse` element is the parent element for information returned to you in response to an `unloadReversal` transaction.

Parent Elements:

[littleOnlineResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [responseTime](#), [message](#)

Optional: [postDate](#), [giftCardResponse](#)

4.402 updateAddOn

The updateAddOn element is the parent of several child elements used to modify an additional charge added to an existing subscription.

Parent Elements:

[updateSubscription](#)

Attributes:

None

Child Elements (all Required):

[addOnCode](#), [name](#), [amount](#), [startDate](#), [endDate](#)

Example: updateAddOn Structure

```
<updateAddOn>
    <addOnCode>Add On Reference Code</addOnCode>
    <name>Name of Add On</name>
    <amount>Amount of Add On</amount>
    <startDate>Start Date of Add On Charge</startDate>
    <endDate>End Date of Add On Charge</endDate>
</updateAddOn>
```

4.403 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>
```

4.404 updateCardValidationNumOnToken

The `updateCardValidationNumOnToken` element is the parent element for the transaction type used to update a CVV2/CVC2/CID code stored temporarily on the platform. You should only use this transaction type if you had previously submitted the account number and security code in a `registerTokenRequest` transaction and now need to change the CVV2/CVC2/CID value.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [littleToken](#), [cardValidationNum](#)

Optional: [orderId](#)

4.405 updateCardValidationNumOnTokenResponse

The updateCardValidationOnTokenResponse element is the parent element for the response to updateCardValidationNumOnToken transactions.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [response](#), [message](#), [responseTime](#)

4.406 updateDiscount

The `updateDiscount` element is the parent of several child elements used to define updates to a discount applied to an existing subscription.

Parent Elements:

[updateSubscription](#)

Attributes:

None

Child Elements (all Required):

`discountCode`, `name`, `amount`, `startDate`, `endDate`

Example: `customerInfo` Structure

```
<updateDiscount>
    <discountCode>Discount Reference Code</discountCode>
    <name>Name of Discount</name>
    <amount>Amount of Discount</amount>
    <startDate>Start Date of Discount</startDate>
    <endDate>End Date of Discount</endDate>
</updateDiscount>
```

4.407 updatePlan

The `updatePlan` element is the parent of the transaction used to activate/deactivate Plans associated with recurring payments. When you deactivate a Plan, you can no longer reference that Plan for use with subscriptions. Existing subscriptions making use of the deactivated Plan will continue to use the Plan until either modified or completed. You can also reactivate a deactivated Plan by updating the Plan and setting the `active` flag to true.

Parent Elements:

`littleOnlineRequest`, `littleRequest`

Attributes:

None

Child Elements:

`planCode`, `active`

Example: `createPlan` Structure

```
<updatePlan>
    <planCode>Plan Reference Code</planCode>
    <active>true or false</active>
</updatePlan>
```

4.408 updatePlanResponse

The updatePlanResponse element is the parent of the response message to the updatePlan transaction used to deactivate Plans associated with recurring payments.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

None

Child Elements:

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

Example: updatePlan Structure

```
<updatePlan>
    <littleTxnId>Transaction ID</littleTxnId>
    <response>Response Reason Code</response>
    <message>Response Message</message>
    <responseTime>Date and Time in GMT</responseTime>
    <planCode>Plan Reference Code</planCode>
</updatePlan>
```

4.409 updateSubscription

The `updateSubscription` element is the parent element for the transaction that updates the subscription information associated with a recurring payment. Using this transaction type you can change the plan, card, billing information, and/or billing date. You can also create, update, or delete a Discount and/or an Add On.

Parent Elements:

[littleOnlineRequest](#), [batchRequest](#)

Attributes:

None

Child Elements:

Required: `subscriptionId`

Optional: `planCode`, `billToAddress`, (choice of) `card`, `paypage`, or `token`, `billingDate`, `createDiscount`, `deleteDiscount`, `updateDiscount`, `createAddOn`, `updateAddOn`, `deleteAddOn`

Example: `updateSubscription` - Change Plan

```
<updateSubscription>
    <subscriptionId>Subscription Id</subscriptionId>
    <planCode>New Plan Code</planCode>
</updateSubscription>
```

Example: `updateSubscription` - Change Card

```
<updateSubscription>
    <subscriptionId>Subscription Id</subscriptionId>
    <card>
        <type>Card Type Abbreviation</type>
        <number>Account Number</number>
        <expDate>Expiration Date</expDate>
    </card>
</updateSubscription>
```

Example: `updateSubscription` - Change Billing Date

```
<updateSubscription>
    <subscriptionId>Subscription Id</subscriptionId>
```

```
<billingDate>New Billing Date</billingDate>
</updateSubscription>
```

Example: updateSubscription - Change Billing Info

```
<updateSubscription>
  <subscriptionId>Subscription Id</subscriptionId>
  <billToAddress>
    <name>Customer's Full Name</name>
    <companyName>Company's Name</companyName>
    <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>
</updateSubscription>
```

Example: updateSubscription - Create Discount

```
<updateSubscription>
  <subscriptionId>Subscription Id</subscriptionId>
  <createDiscount>
    <discountCode>Discount Reference Code</discountCode>
    <name>Name of Discount</name>
    <amount>Amount of Discount</amount>
    <startDate>Start Date of Discount</startDate>
    <endDate>End Date of Discount</endDate>
  </createDiscount>
</updateSubscription>
```

Example: updateSubscription - Create Add On

```
<updateSubscription>
  <subscriptionId>Subscription Id</subscriptionId>
```

```

<createAddOn>
  <addOnCode>Add On Reference Code</addOnCode>
  <name>Name of Add On</name>
  <amount>Amount of Add On</amount>
  <startDate>Start Date of Add On Charge</startDate>
  <endDate>End Date of Add On Charge</endDate>
</createAddOn>
</updateSubscription>

```

Example: updateSubscription - Update Discount

```

<updateSubscription>
  <subscriptionId>Subscription Id</subscriptionId>
  <updateDiscount>
    <discountCode>Discount Reference Code</discountCode>
    <name>Name of Discount</name>
    <amount>Amount of Discount</amount>
    <startDate>Start Date of Discount</startDate>
    <endDate>End Date of Discount</endDate>
  </updateDiscount>
</updateSubscription>

```

Example: updateSubscription - Update Add On

```

<updateSubscription>
  <subscriptionId>Subscription Id</subscriptionId>
  <updateAddOn>
    <addOnCode>Add On Reference Code</addOnCode>
    <name>Name of Add On</name>
    <amount>Amount of Add On</amount>
    <startDate>Start Date of Add On Charge</startDate>
    <endDate>End Date of Add On Charge</endDate>
  </updateAddOn>
</updateSubscription>

```

Example: updateSubscription - Delete Discount

```

<updateSubscription>
  <subscriptionId>Subscription Id</subscriptionId>

```

```
<deleteDiscount>
    <discountCode>Discount Reference Code</discountCode>
</deleteDiscount>
</updateSubscription>
```

Example: updateSubscription - Delete Add On

```
<updateSubscription>
    <subscriptionId>Subscription Id</subscriptionId>
    <deleteAddOn>
        <addOnCode>Add On Reference Code</addOnCode>
    </deleteAddOn>
</updateSubscription>
```

4.410 updateSubscriptionResponse

The updateSubscriptionresponse element is the parent element for the response to an updateSubscription transaction.

Parent Elements:

[littleOnlineResponse](#), [batchResponse](#)

Attributes:

None

Child Elements:

[subscriptionId](#), [littleTxnId](#), [response](#), [message](#), [responseTime](#), [tokenResponse](#)

4.411 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>
```

4.412 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 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

4.413 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

4.414 vendorCredit

The `vendorCredit` element is the parent element for the transaction type that a PayFac uses to move funds from the PayFac Settlement Account the Vendor Account.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [accountInfo](#), [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#), [vendorName](#)

4.415 vendorCreditResponse

The `vendorCreditResponse` element is the parent element for information returned to you in response to a `vendorCredit` transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.416 vendorDebit

The `vendorDebit` element is the parent element for the transaction type that a PayFac uses to move funds from the Vendor Account to the PayFac Settlement Account.

Parent Elements:

[batchRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	String	Yes	For PayFacs, this attribute does not segregate transactions in iQ. This field does appear in various SSR reports. minLength = 1 maxLength = 25

Child Elements:

Required: [accountInfo](#), [amount](#), [fundingSubmerchantId](#), [fundsTransferId](#), [vendorName](#)

4.417 vendorDebitResponse

The vendorDebitResponse element is the parent element for information returned to you in response to a vendorDebit transaction.

Parent Elements:

[batchResponse](#)

Attributes:

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

Child Elements:

Required: [littleTxnId](#), [fundsTransferId](#), [response](#), [responseTime](#), [message](#)

4.418 vendorName

The vendorName element is a required child of both the vendorCredit and vendorDebit elements and specifies the name of the vendor involved in the funding instructions.

Type = String; **minLength** = 1; **maxLength** = 256

Parent Elements:

[vendorCredit](#), [vendorDebit](#)

Attributes:

None

Child Elements:

None

4.419 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

4.420 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

4.421 version

The **version** element is a required child of the `applepay` element and provides version information about the payment token.

Type = String; **minLength** = 5; **maxLength** = 20

Parent Elements:

[applepay](#)

Attributes:

None

Child Elements:

None

4.422 virtualAccountNumber

The `virtualAccountNumber` element is an optional child of the `enhancedAuthResponse` element and indicates if the card number used for the transaction corresponds to a virtual account number.

Type = Boolean; **Valid Values** = true or false

Parent Elements:

`enhancedAuthResponse`

Attributes:

None

Child Elements:

None

4.423 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

4.424 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

4.425 virtualGiftCard

The `virtualGiftCard` element is an optional child of the `activate` transaction. You include this element when you are requesting a Virtual Gift Card.

NOTE: In an early iteration of schema V8.22 issued in September of 2013 the `accountNumberLength` and `giftCardBin` child elements were defined as optional. If you coded to the earlier version, be aware that these elements are now required children of `virtualGiftCard`. If you do not include these elements, the transaction will fail XML validation.

Parent Elements:

`activate`

Attributes:

None

Child Elements (all required):

`accountNumberLength`, `giftCardBin`

Example: `virtualGiftCard` Structure

```
<virtualGiftCard>
    <accountNumberLength>Length of Virtual Card Number</accountNumberLength>
    <giftCardBin>Requested BIN of Virtual Gift Card</giftCardBin>
</virtualGiftCard>
```

4.426 virtualGiftCardBin

The VirtualGiftCardBin element is an optional child of the activateReversal element defining the BIN of the virtual Gift Card.

Type = String; **minLength** = N/A; **maxLength** = 10

Parent Elements:

[activateReversal](#)

Attributes:

None

Child Elements:

None

4.427 virtualGiftCardResponse

The `virtualGiftCardResponse` element is an optional child of the `activateResponse` transaction. This element is returned when you request a Virtual Gift Card number via an `activate` transaction and through its children, defines the virtual gift Card number, as well as the Card Validation number.

Parent Elements:

[activateResponse](#)

Attributes:

None

Child Elements (all optional):

[accountNumber](#), [cardValidationNum](#), [pin](#)

Example: `virtualGiftCardResponse` Structure

```
<virtualGiftCardResponse>
    <accountNumber>Virtual Card Number</accountNumber>
    <cardValidationNum>Validation Number</cardValidationNum>
    <pin>Pin Number</pin>
</virtualGiftCardResponse>
```

4.428 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

4.429 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 Recycling Engine, you can use the `void` transaction to halt the recycling of a `sale` transaction.

Parent Elements:

[littleOnlineRequest](#)

Attributes:

Attribute Name	Type	Required?	Description
<code>id</code>	String	Yes	A unique identifier assigned by the presenter and mirrored back in the response. minLength = 1 maxLength = 25
<code>customerId</code>	String	No	A value assigned by the merchant to identify the consumer. minLength = N/A maxLength = 50
<code>reportGroup</code>	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 10 for additional information. minLength = 1 maxLength = 25

Child Elements:

Required: [littleTxnId](#)

Optional: [processingInstructions](#)

4.430 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	Yes	The response returns the same value submitted in the void transaction. minLength = 1 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

Child Elements: (Required)

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

Child Elements: (Optional)

[recycling](#)

4.431 wallet

The `wallet` element is an optional child of both the `authorization` and `sale` transactions. You must use this element along with its child elements, when the consumer uses MasterPass or Visa Checkout to make a purchase.

Parent Elements:

[authorization](#), [sale](#)

Attributes:

None

Child Elements:

[walletSourceType](#), [walletSourceTypeId](#)

Example: wallet Structure

```
<wallet>
    <walletSourceType>MasterPass or VisaCheckout</walletSourceType>
    <walletSourceTypeId>MasterPass ID or VCIND</walletSourceTypeId>
</wallet>
```

4.432 walletSourceType

The `walletSourceType` element is a required child of the `wallet` element, which defines the source of the transaction information. You must submitted this element with the transaction when the consumer uses MasterPass or Visa Checkout.

Type = String (Enum); **minLength** = N/A; **maxLength** = N/A

Parent Elements:

[wallet](#)

Attributes:

None

Child Elements:

None

Enumerations:

Enumeration	Description
MasterPass	The origin of the information used in the transaction is from MasterPass.
VisaCheckout	The origin of the information used in the transaction is from Visa Checkout.

4.433 walletSourceTypeId

The `walletSourceTypeId` element is a required child of the `wallet` element. For MasterPass transactions, the value of this element is returned from MasterPass. For Visa Checkout transactions, set this value to VCIND.

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

Parent Elements:

[wallet](#)

Attributes:

None

Child Elements:

None

4.434 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 PayPal Credit transactions.

NOTE:

As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = Integer; **totalDigits** = 2

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.435 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 PayPal Credit transactions.

NOTE: As of September, 2016, the Vantiv eCommerce platform no longer supports PayPal Credit for new merchants or existing merchants not already using this payment method. Existing merchants already using PayPal Credit should consult their Vantiv Relationship Manager.

Type = Integer; totalDigits = 2

Parent Elements:

[customerInfo](#)

Attributes:

None

Child Elements:

None

4.436 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 `zip` element as 20 characters, in practice you should never exceed 10 characters in your submissions.

If including the `zip` element for eCheck Verification, do not exceed 9 characters and do not use dashes.

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](#)
- [3DS Authentication Result Codes](#)
- [AVS Response Codes](#)
- [AAVS Response Codes](#)
- [Card Validation Response Codes](#)
- [Advanced Fraud Tools Triggered Rules](#)
- [XML Validation Error Messages](#)
- [Additional Response Header Error Messages](#)
- [ACH Return Reason Codes](#)
- [ACH NoC Change Codes](#)
- [Canadian eCheck Return 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
001	Transaction Received	Info	This is sent to acknowledge that the submitted transaction has been received.
000	Approved	Approved	No action required.
010	Partially Approved	Approved	The authorized amount is less than the requested amount.
011	Offline Approval	Approved	Offline approval issued while the terminal is unable to communicate with the issuer.
013	Offline Approval (unable to go online)	Approved	Offline approval issued while the terminal is unable to communicate with the issuer.
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.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
108	Try again later	Soft	Returned if the eProtect token is not immediately available, when submitting an Auth or Sale 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. Appears in Declined Transaction report.
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.
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.
150	Original transaction found.	Info	A Query transaction response indicating that the original transaction was found.
151	Original transaction not found.	Info	A Query transaction response indicating that the original transaction was not found.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
152	Original transaction found, but response not yet available.	Info	A Query transaction response indicating that the original transaction was found, but the final response information is not yet available.
153	Query transaction not enabled.	Info	A Query transaction response indicating that you are not enabled for use of the Query transaction.
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 Relationship Manager.
192	Merchant not certified/enabled for IIAS	Hard Decline	Your organization is not certified or enabled for IIAS/FSA transactions.
206	Issuer Generated Error	Soft Decline	An unspecified error was returned by the issuer. Please retry the transaction and if the problem persist, contact the issuing bank.
207	Pickup card - Other than Lost/Stolen	Hard Decline	The issuer indicated that the gift card should be removed from use.
209	Invalid Amount	Hard Decline	The specified amount is invalid for this transaction.
211	Reversal Unsuccessful	Hard Decline	The reversal transaction was unsuccessful.
212	Missing Data	Hard Decline	Contact your Relationship Manager.
213	Pickup Card - Lost Card	Hard Decline	The submitted card was reported as lost and should be removed from use.
214	Pickup Card - Stolen Card	Hard Decline	The submitted card was reported as stolen and should be removed from use.
215	Restricted Card	Hard Decline	The specified Gift Card is not available for use.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
216	Invalid Deactivate	Hard Decline	The Deactivate transaction is invalid for the specified card.
217	Card Already Active	Hard Decline	The submitted card is already active.
218	Card Not Active	Hard Decline	The submitted card has not been activated.
219	Card Already Deactivate	Hard Decline	The submitted card has already been deactivated.
221	Over Max Balance	Hard Decline	The activate or load amount exceeds the maximum allowed for the specified gift Card.
222	Invalid Activate	Hard Decline	The activate transaction is not valid or can no longer be reversed.
223	No transaction Found for Reversal	Hard Decline	The transaction referenced in the reversal transaction does not exist.
226	Incorrect CVV	Hard Decline	The transaction was declined because it was submitted with the incorrect security code.
229	Illegal Transaction	Hard Decline	The transaction would violate the law.
251	Duplicate Transaction	Hard Decline	The transaction is a duplicate of a previously submitted transaction. Appears in Declined Transaction report.
252	System Error	Hard Decline	Contact your Relationship Manager.
253	Deconverted BIN	Hard Decline	The BIN is no longer valid.
254	Merchant Depleted	Hard Decline	No balance remains on gift Card.
255	Gift Card Escheated	Hard Decline	The Gift Card has been seized by the government while resolving an estate.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
256	Invalid Reversal Type for Credit Card Transaction	Hard Decline	You attempted to use a Closed Loop Gift Card reversal transaction to reverse a credit card transaction. For example, you cannot use a Deposit Reversal transaction to reverse a Capture. To reverse a credit card Capture transaction, use a Credit transaction.
257	System Error (message format error)	Hard Decline	Issuer reported message format is incorrect. Contact your Relationship Manager.
258	System Error (cannot process)	Hard Decline	Issuer reported transaction could not be processed. Contact your Relationship Manager.
271	Refund rejected due to pending deposit status	Soft Decline	The refund is tied to a deposit that is still in pending state or the state is in doubt. You can retry the refund at a later time.
272	Refund rejected due to declined deposit status	Hard Decline	The refund is tied to a deposit that failed.
273	Refund rejected by the processing network	Soft Decline	The refund is tied to a deposit that succeeded, but was declined by PayPro.
284	Capture, Credit and AuthReversal tags cannot be used for Gift Card Transactions	Hard Decline	You must use the Gift Card version of these transactions for Gift Cards (i.e., giftCardCapture, giftCardCredit, and giftCardAuthReversal).
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).

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
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.
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. Appears in Declined Transaction report.
307	Restricted Card	Hard Decline	The card has a restriction preventing approval for this transaction. Please contact the issuing bank for a specific reason. You may also receive this code if the transaction was declined due to Prior Fraud Advice Filtering and you are using a schema version V8.10 or older.
308	Restricted Card - Chargeback	Hard Decline	This transaction is being declined due the operation of the Prior Chargeback Card Filtering Service or 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 Prepaid Card Filtering service.
310	Invalid track data	Hard Decline	The track data is not valid.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
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. Appears in Declined Transaction report.
312	Restricted Card - International Card Filtering Service	Hard Decline	This transaction is being declined due the operation of the International Card Filtering Service.
313	International filtering for issuing card country <country> (where <country> is the 3-character country code)	Hard Decline	This is returned when the transaction involves a US based merchant processing Canadian transactions has a transaction that uses a US card.
315	Restricted Card - Auth Fraud Velocity Filtering Service	Hard Decline	This transaction is being declined due the operation of the 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 Fraud Chargeback Prevention Service (FCPS). Appears in Declined Transaction report.
318	Restricted Card - Auth Fraud Advice Filtering Service	Hard Decline	This transaction is being declined due the operation of the Auth Fraud Advice Filtering Service.
319	Restricted Card - Fraud AVS Filtering Service	Hard Decline	This transaction is being declined due the operation of the Auth Fraud AVS Filtering Service.
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).

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
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. If you are enabled for Velocity Fraud Filtering, but have not upgraded to V8.9, you will receive this code for filtered transactions. If you are enabled for AVS Fraud Filtering, but have not upgraded to V8.13, 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. Appears in Declined Transaction report
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.
331	Invalid POS Capability for Cardholder Authorized Terminal Transaction	Hard Decline	For a Cardholder Authorized Terminal Transaction the POS capability must be set to magstripe.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
332	Invalid POS Cardholder ID for Cardholder Authorized Terminal Transaction	Hard Decline	For a Cardholder Authorized Terminal Transaction the POS Cardholder ID must be set to nopin.
335	This method of payment does not support authorization reversals	Hard Decline	You can not perform an Authorization Reversal transaction for this payment type.
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.
337	Transaction did not convert to Pinless	Soft Decline	Retry the transaction.
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. Appears in Declined Transaction report.
347	Invalid billing descriptor	Hard Decline	The billing descriptor is not valid because you are not authorized to send transactions with custom billing fields. Appears in Declined Transaction report.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
348	Invalid Report Group	Hard Decline	The Report Group specified in the transaction is invalid, because it is either not in the defined list of acceptable Report Groups or there is a mis-match between the Report Group and the defined Billing Descriptor.
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.
357	Missing healthcareIIS tag for an FSA transaction	Hard Decline	The FSA Transactions submitted does not contain the <healtcareIIS> data element.
358	Restricted by Little due to security code mismatch.	Hard Decline	The transaction was declined due to the security code (CVV2, CID, etc) not matching.
360	No transaction found with specified littleTxnId	Hard Decline	There were no transactions found with the specified littleTxnId. Appears in Declined Transaction report.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
361	Authorization no longer available	Hard Decline	<p>The authorization for this transaction is no longer available. Either the authorization has already been consumed by another capture, or the authorization has expired.</p> <p>Appears in Declined Transaction report.</p>
362	Transaction Not Voided - Already Settled	Hard Decline	<p>This transaction cannot be voided; it has already been delivered.</p> <p>Appears in Declined Transaction report.</p>
363	Auto-void on refund	Hard Decline	<p>This transaction (both capture and refund) has been voided.</p> <p>Appears in Declined Transaction report.</p>
364	Invalid Account Number - original or NOC updated eCheck account required	Hard Decline	<p>The submitted account number is invalid. Confirm the original account number or check NOC for new account number.</p>
365	Total credit amount exceeds capture amount	Hard Decline	<p>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.</p> <p>Appears in Declined Transaction report.</p>
366	Exceed the threshold for sending redeposits	Hard Decline	<p>NACHA rules allow two redeposit attempts within 180 days of the settlement date of the initial deposit attempt. This threshold has been exceeded.</p> <p>Appears in Declined Transaction report.</p>

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
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. Appears in Declined Transaction report.
368	Invalid check number	Soft Decline	The check number is invalid.
369	Redeposit against invalid transaction type	Hard Decline	The redeposit attempted against an invalid transaction type. Appears in Declined Transaction report.
370	Internal System Error - Call Little	Hard Decline	There is a problem with the system. Contact eCommerceSupport@vantiv.com.
371	Original Transaction has been Processed - Future Redeposits Canceled	Hard Decline	Do not send additional redeposit transactions, since the original transaction was processed. Appears in Declined Transaction report.
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.
375	Merchant is not enabled for surcharging	Hard Decline	The submitted transaction contained a surcharge and the merchant is not enabled for surcharging.
376	This method of payment does not support surcharging	Hard Decline	The use of a surcharge is only allowed for Visa and MasterCard methods of payment.
377	Surcharge is not valid for debit or prepaid cards	Hard Decline	You cannot apply a surcharge to a transaction using a debit or prepaid card.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
378	Surcharge cannot exceed 4% of the sale amount	Hard Decline	The surcharge in the submitted transaction exceeded 4% maximum allowed for a surcharge.
380	Secondary amount cannot exceed the sale amount	Hard Decline	The secondary amount exceeded the sale amount in the submitted transaction.
381	This method of payment does not support secondary amount	Hard Decline	The submitted method of payment does not allow the use of Convenience Fees.
382	Secondary amount cannot be less than zero	Hard Decline	The secondary amount must be a positive integer.
383	Partial transaction is not supported when including a secondary amount	Hard Decline	Transactions set to allow partial authorizations cannot include a secondary amount.
384	Secondary amount required on partial refund when used on deposit	Hard Decline	If the associated sale or capture transaction included a secondary amount, an associated partial refund must include a secondary amount.
385	Secondary amount not allowed on refund if not included on deposit	Hard Decline	If the associated sale or capture transaction did not include a secondary amount, you cannot include a secondary amount on an associated refund.
401	Invalid E-mail	Hard Decline	The e-mail address provided is not valid. Verify that it was entered correctly.
469	Invalid Recurring Request - See Recurring Response for Details	Hard Decline	The Recurring Request was invalid, which invalidated the transaction. The Response Code and Message in the Recurring Response contains additional information.
470	Approved - Recurring Subscription Created	Approved	The recurring request was processed successfully.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
471	Parent Transaction Declined - Recurring Subscription Not Created	Hard Decline	The original payment transaction was declined, so the recurring payments have not been scheduled. Appears in Declined Transaction report.
472	Invalid Plan Code	Hard Decline	The plan specified in the recurring request was invalid.
473	Scheduled Recurring Payment Processed	Approved	The scheduled recurring payment has been processed successfully.
475	Invalid Subscription Id	Hard Decline	The referenced subscription Id does not exist. Appears in Declined Transaction report.
476	Add On Code Already Exists	Hard Decline	The specified Add On code already exists. Appears in Declined Transaction report.
477	Duplicate Add On Codes in Requests	Hard Decline	Multiple createAddOn requests submitted with the same Add On Code.
478	No Matching Add On Code for the Subscription	Hard Decline	The Add On code specified does not exist. Appears in Declined Transaction report.
480	No Matching Discount Code for the Subscription	Hard Decline	The Discount Code supplied in the updateDiscount or deleteDiscount transaction does not exist. Appears in Declined Transaction report.
481	Duplicate Discount Codes in Request	Hard Decline	Multiple createDiscount requests submitted with the same Discount Code.
482	Invalid Start Date	Hard Decline	The supplied Start Date is invalid.
483	Merchant Not Registered for Recurring Engine	Hard Decline	You are not registered for the use of the Recurring Engine.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
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.
521	Soft Decline - Card reader decryption service is not available	Soft Decline	The connection to the decryption service is currently unavailable. Please retry the transaction and/or contact your Relationship Manager.
523	Soft Decline - Decryption failed	Soft Decline	Our attempt to decrypt the card information failed. Please retry the transaction.
524	Hard Decline - Input data is invalid.	Hard Decline	The submitted data is invalid.
530	Apple Pay Key Mismatch	Hard Decline	The submitted publicKeyHash element does not match any configured entries. Contact your Implementation Consultant.
531	Apple Pay Decryption Failed	Hard Decline	Vantiv was unable to decrypt the submitted information.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
550	Restricted Device or IP - ThreatMetrix Fraud Score Below Threshold	Hard Decline	The transaction was declined because the resulting ThreatMetrix Fraud Score was below the acceptable threshold set in the merchant's policy.
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.).
NOTE: The Response Message associated with Response Code 602 is inaccurate due to a remapping of PayPal Response Codes. Please read the Description below for the recommended action when receiving Response Code 602.			
602	Soft Decline - Buyer has alternate funding source	Soft Decline	<p>The transaction could not be completed for one of the following reasons:</p> <ul style="list-style-type: none"> • The billing address associated with the financial Instrument could not be confirmed. • The transaction exceeds the card limit. • The transaction was denied by the card issuer. <p>You should establish error handling logic that directs the customer to contact PayPal to resolve the issue with their account.</p>
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.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
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 eCommerceSupport@vantiv.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.
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 eCommerceSupport@vantiv.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.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
621	Hard Decline - required PayPal parameter missing	Hard Decline	A PayPal response indicating missing parameters are required. Contact eCommerceSupport@vantiv.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 eCommerceSupport@vantiv.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.
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 your Relationship Manager to adjust your PayPal merchant profile preferences.
628	Internal System Error with PayPal - Contact Little	Hard Decline	There is a problem with your username and password. Contact eCommerceSupport@vantiv.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.
637	Invalid terminal Id	Hard Decline	The terminal Id submitted with the POS transaction is invalid.
701	Under 18 years old	Hard Decline	A PayPal Credit response indicating the customer is under 18 years of age based upon the date of birth.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
702	Bill to outside USA	Hard Decline	A PayPal Credit response indicating the billing address is outside the United States.
703	Bill to address is not equal to ship to address	Hard Decline	A PayPal Credit response indicating that the billing address does not match the shipping address.
704	Declined, foreign currency, must be USD	Hard Decline	A PayPal Credit response indicating the transaction is declined, because it is not in US dollars.
705	On negative file	Hard Decline	A PayPal Credit response indicating the account is on the negative file.
706	Blocked agreement	Hard Decline	A PayPal Credit response indicating a blocked agreement account status.
707	Insufficient buying power	Other	A PayPal Credit response indicating that the account holder does not have sufficient credit available for the transaction amount.
708	Invalid Data	Hard Decline	A PayPal Credit response indicating that there are one or more problems with the submitted data.
709	Invalid Data - data elements missing	Hard Decline	A PayPal Credit response indicating one or more required data elements are missing. Also, returned for an eCheck transaction that is missing a required data element. For example, failure to include the name element in an echeckSale or echeckCredit transaction would result in this code being returned.
710	Invalid Data - data format error	Hard Decline	A PayPal Credit response indicating that some data was formatted incorrectly.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
711	Invalid Data - Invalid T&C version	Hard Decline	A PayPal Credit response indicating the T&C version is invalid.
712	Duplicate transaction	Hard Decline-	A PayPal Credit response indicating that the transaction is a duplicate.
713	Verify billing address	Hard Decline	A PayPal Credit response indicating that you should verify the billing address.
714	Inactive Account	Hard Decline	A PayPal Credit response indicating the customer account is inactive.
716	Invalid Auth	Hard Decline	A PayPal Credit response indicating that the referenced authorization is invalid.
717	Authorization already exists for the order	Hard Decline	A PayPal Credit 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.
805	Card Validation Number Updated	Approved	The stored value for CVV2/CVC2/CID has been successfully updated.
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.
823	Token Invalid	Hard Decline	The submitted token is invalid.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
835	Capture amount can not be more than authorized amount	Hard Decline	The amount in the submitted Capture exceeds 115% of the authorized amount. Appears in Declined Transaction report.
850	Tax Billing only allowed for MCC 9311	Hard Decline	Tax Billing elements are allowed only for MCC 9311.
851	MCC 9311 requires taxType element	Hard Decline	Missing taxType element
852	Debt Repayment only allowed for VI transactions on MCCs 6012 and 6051	Hard Decline	You must be either MCC 6012 or 6051 to designate a Visa transaction as Debt Repayment (debtRepayment element set to true).
861	Routing Number did not match one on file for token	Soft Decline	The routing number submitted does not match the number submitted when the token was created. Verify the routing number and resubmit the transaction.
877	Invalid Pay Page Registration Id	Hard Decline	An eProtect response indicating that the Registration ID submitted is invalid.
878	Expired Pay Page Registration Id	Hard Decline	An eProtect response indicating that the Registration ID has expired (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 eProtect.
898	Generic token registration error	Soft Decline	There is an unspecified token registration error; contact your Relationship Manager
899	Generic token use error	Soft Decline	There is an unspecified token use error; contact your Relationship Manager.
900	Invalid Bank Routing Number	Hard Decline	The eCheck routing number submitted with this transaction has failed validation.
901	Missing Name	Hard Decline	The customer name is required for SEPA transactions.

TABLE A-1 Valid Values for the Response and Message Elements (Continued)

Response Code	Response Message	Response Type	Description
902	Invalid Name	Hard Decline	The customer name must be a minimum of two characters for SEPA transactions.
903	Missing Billing Country Code	Hard Decline	The Billing Country code is required for SEPA transactions.
904	Invalid IBAN	Hard Decline	The submitted International Bank Account number is invalid. Please correct the number and resubmit the transaction.
905	Missing Email Address	Hard Decline	The customer email address is required for SEPA transactions.
917	Invalid Billing Country	Hard Decline	Some of the alternative payment methods restrict the allowed consumer's billing country codes. For example, iDEAL only allows country code NL.
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 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 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 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.
959	Phone number required for eCheck Verifications	Hard Decline	An eCheck response indicating that the phone number of the customer is required for eCheck verifications.
961	Card Brand token not supported	Hard Decline	This code is returned if the merchant submits a Visa generated token.
962	Private Label Card not supported	Hard Decline	This code is returned if the transaction involves a Visa Private Label card.

A.2 3DS Authentication Result Codes

Table A-2 contains a list of valid authentication result codes returned by Visa for the Verified by Visa service or MasterCard for the MasterCard SecureCode service. It specifies what authentication result values apply to what order sources.

NOTE: The Discover ProtectBuy service only returns one of the following Authentication Result Codes: 0, 1, or 2.

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

TABLE A-2 Authentication Result Codes (Continued)

Authentication Result Code	Description
9	CAVV failed verification; Visa generated CAVV because Issuer ACS was not available.
A	CAVV passed verification; Visa generated CAVV because Issuer Access Control Server (ACS) was not available.
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: For American Express transactions, if the submitted security code does not match, the transaction is declined with a Response Reason Code of 352 - Decline CVV2/CID Fail.

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 Advanced Fraud Tools Triggered Rules

This section provides definitions of the triggered rules returned in the Advanced Fraud Results (`advancedFraudResults` element) section of the response message (see Example below). ThreatMetrix uses the rules triggered by each advanced fraud check to determine the device reputation score, which in turn determines the final review status: Pass, Review, or Fail.

NOTE: The rules/descriptions in this document reflect those used in the generic merchant policy. Depending upon the policy configured in your merchant profile, some rules may not apply to you, or additional rules, not defined here, may appear in your results.

Example: `advancedFraudResults` Structure

```
<advancedFraudResults>
    <deviceReviewStatus>pass, fail, review, etc.</deviceReviewStatus>
    <deviceReputationScore>Score Returned from
    ThreatMetrix</deviceReputationScore>
    <triggeredRule>Triggered Rule #1</triggeredRule>
    .
    .
    .
    <triggeredRule>Triggered Rule #N</triggeredRule>
</advancedFraudResults>
```

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
10PaymentsOnDeviceLocalDay	This device submitted 10 or more payments in the previous 24 hours. This is atypical and may be an indicator of misuse.
10PaymentsOnDeviceLocalHour	This device has submitted 10 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
10PaymentsOnFuzzyDeviceLocalDay	This device appears to have submitted 10 or more payments in the previous 24 hours. This is atypical and may be an indicator of misuse.
10PaymentsOnFuzzyDeviceLocalHour	This device appears to have submitted 10 or more payments in the previous hour. This is atypical and may be an indicator of misuse.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
10PaymentsOnTrueIPLocalDay	This True IP submitted 10 or more payments in the previous day. This is atypical and may be an indicator of misuse.
10PaymentsOnTrueIPLocalHour	This True IP submitted 10 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
10PaymentsWithEmailAddressLocalDay	This email address submitted 10 or more payments in the previous day. This is atypical and may be an indicator of misuse.
15PaymentsWithCustomerIDLocalDay	This customer ID submitted 15 or more payments in the previous day. This is atypical and may be an indicator of misuse.
15PaymentsWithPaymentCardLocalDay	This payment card submitted 15 or more payments in the previous day. This is atypical and may be an indicator of misuse.
100PaymentsOnDeviceLocalMonth	This device submitted 100 or more payments in the previous month. This is atypical and may be an indicator of misuse.
100PaymentsOnFuzzyDeviceLocalMonth	This device appears to have submitted 100 or more payments in the previous month. This is atypical and may be an indicator of misuse.
100PaymentsOnTrueIPLocalMonth	This True IP submitted 100 or more payments in the previous month. This is atypical and may be an indicator of misuse.
2ScreenResolutionsPerDeviceGlobalDay	This device used 2 or more screen resolutions in the past day. This is atypical and may indicate misuse.
20PaymentsOnDeviceLocalDay	This device submitted 20 or more payments in the previous 24 hours. This is atypical and may be an indicator of misuse.
20PaymentsOnFuzzyDeviceLocalDay	This device appears to have submitted 20 or more payments in the previous 24 hours. This is atypical and may be an indicator of misuse.
20PaymentsOnTrueIPLocalDay	This True IP submitted 20 or more payments in the previous day. This is atypical and may be an indicator of misuse.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
20PaymentsWithCustomerIDLocalWeek	This customer ID submitted 20 or more payments in the previous week. This is atypical and may be an indicator of misuse.
20PaymentsWithPaymentCardLocalWeek	This payment card submitted 20 or more payments in the previous week. This is atypical and may be an indicator of misuse.
3CustomerIDsPerDeviceLocalDay	This device submitted transactions using 3 or more distinct customer IDs in the previous 24 hours. This is abnormal and may be an indicator of a card-testing attack or free-trial abuse.
3CustomerIDsPerDeviceLocalWeek	This device submitted transactions using 3 or more distinct customer IDs in the previous 7 days. This is abnormal and may be an indicator of a card-testing attack or free-trial abuse.
3DevicesPerCustomerIDLocalDay	Three or more devices have been used to submit transactions with this customer ID in the previous 24 hours. This may be an indicator of misuse.
3DevicesPerCustomerIDLocalWeek	Three or more devices have been used to submit transactions with this customer ID in the previous 7 days. This may be an indicator of misuse.
3DevicesPerEmailGlobalDay	Three or more devices have been used to submit transactions with this email address in the previous 24 hours. This may be an indicator of misuse.
3DevicesPerEmailGlobalWeek	Three or more devices have been used to submit transactions with this email address in the previous 7 days. This may be an indicator of misuse.
3DevicesPerPaymentCardGlobalDay	Three or more devices have been used to submit transactions with this payment card in the previous 24 hours. This may be an indicator of misuse.
3DevicesPerPaymentCardGlobalWeek	Three or more devices have been used to submit transactions with this payment card in the previous 7 days. This may be an indicator of misuse.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
3EmailsPerDeviceGlobalDay	This device has submitted transactions using 3 or more distinct email addresses in the previous 24 hours. This is abnormal and may be an indicator of a card-testing attack or free-trial abuse.
3EmailsPerDeviceGlobalWeek	This device has submitted transactions using 3 or more distinct email addresses in the previous 7 days. This is abnormal and may be an indicator of a card-testing attack or free-trial abuse.
3EmailsPerFuzzyDeviceLocalHour	This device appears to have submitted transactions using 3 or more distinct email addresses in the previous 60 minutes. This is abnormal and may be an indicator of a card-testing attack or free-trial abuse.
3PaymentCardsPerDeviceGlobalDay	This device has submitted transactions using 3 or more distinct payment cards in the previous 24 hours. This is abnormal and may be an indicator of a card-testing attack or free-trial abuse.
3PaymentCardsPerDeviceGlobalWeek	This device has submitted transactions using 3 or more distinct payment cards in the previous 7 days. This is abnormal and may be an indicator of a card-testing attack or free-trial abuse.
3PaymentsOnDeviceLocalHour	This device has submitted 3 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
3PaymentsOnFuzzyDeviceLocalHour	This device appears to have submitted 3 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
3PaymentsOnTrueIPLocalHour	This True IP submitted 3 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
3ProxiesPerDeviceGlobalDay	This device submitted transactions through 3 or more distinct IP proxies in the previous 24 hours. This is atypical and may be an indicator of misuse.
4ScreenResolutionsPerDeviceGlobalDay	This device has used 4 or more screen resolutions in the past day. This is atypical and may indicate misuse.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
5PaymentsOnDeviceLocalDay	This device has submitted 5 or more payments in the previous 24 hours. This is atypical and may be an indicator of misuse.
5PaymentsOnDeviceLocalHour	This device has submitted 5 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
5PaymentsOnFuzzyDeviceLocalDay	This device appears to have submitted 5 or more payments in the previous 24 hours. This is atypical and may be an indicator of misuse.
5PaymentsOnFuzzyDeviceLocalHour	This device appears to have submitted 5 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
5PaymentsOnTrueIPLocalDay	This True IP has submitted 5 or more payments in the previous day. This is atypical and may be an indicator of misuse.
5PaymentsOnTrueIPLocalHour	This True IP submitted 5 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
5PaymentsWithCustomerIDLocalHour	This customer ID submitted 5 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
5PaymentsWithEmailAddressLocalHour	This email address has submitted 5 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
5PaymentsWithPaymentCardLocalHour	This payment card submitted 5 or more payments in the previous hour. This is atypical and may be an indicator of misuse.
50PaymentsOnDeviceLocalWeek	This device has submitted 50 or more payments in the previous week. This is atypical and may be an indicator of misuse.
50PaymentsOnFuzzyDeviceLocalWeek	This device appears to have submitted 50 or more payments in the previous week. This is atypical and may be an indicator of misuse.
50PaymentsOnTrueIPLocalMonth	This True IP has submitted 50 or more payments in the previous month. This is atypical and may be an indicator of misuse.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
20PaymentsWithCustomerIDLocalMonth	This customer ID submitted 20 or more payments in the previous month. This is atypical and may be an indicator of misuse.
20PaymentsWithPaymentCardLocalWeek	This payment card submitted 20 or more payments in the previous week. This is atypical and may be an indicator of misuse.
AnonymousProxy	This transaction was submitted through an anonymous web proxy, a method that is sometimes employed when trying to cloak one's identity.
AnonymousProxyIP	This transaction was submitted through an anonymous proxy IP Address, a method that is sometimes employed when trying to cloak one's identity.
BINCustomerAddressGeolocationMismatch	The customer's bill-to address country does not match that of the payment card's issuing bank. This may be an indicator of a fraud attack.
ComputerGeneratedEmail	This email address may have been automatically generated by a computer. Fraudsters frequently employ automated bots that create email addresses programmatically to enable their fraud attacks.
CookiesDisabled	The browser used to submit this transaction has disabled cookies. This is common to fraud attacks and may be an indicator of misuse.
CookiesJavascriptDisabled	The browser used to submit this transaction has disabled cookies and JavaScript. This is common to fraud attacks and may be an indicator of misuse.
CustomAttribute1OnLocalBlacklist	Custom attribute 1 for this transaction is on Vantiv's ThreatMetrix-hosted blacklist.
CustomAttribute1OnLocalWhitelist	Custom attribute 1 for this transaction is on Vantiv's ThreatMetrix-hosted whitelist.
CustomAttribute2OnLocalBlacklist	Custom attribute 2 for this transaction is on Vantiv's ThreatMetrix-hosted blacklist.
CustomAttribute2OnLocalWhitelist	Custom attribute 2 for this transaction is on Vantiv's ThreatMetrix-hosted whitelist.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
CustomAttribute3OnLocalBlacklist	Custom attribute 3 for this transaction is on Vantiv's ThreatMetrix-hosted blacklist.
CustomAttribute3OnLocalWhitelist	Custom attribute 3 for this transaction is on Vantiv's ThreatMetrix-hosted whitelist.
CustomAttribute4OnLocalBlacklist	Custom attribute 4 for this transaction is on Vantiv's ThreatMetrix-hosted blacklist.
CustomAttribute4OnLocalWhitelist	Custom attribute 4 for this transaction is on Vantiv's ThreatMetrix-hosted whitelist.
CustomAttribute5OnLocalBlacklist	Custom attribute 5 for this transaction is on Vantiv's ThreatMetrix-hosted blacklist.
CustomAttribute5OnLocalWhitelist	Custom attribute 5 for this transaction is on Vantiv's ThreatMetrix-hosted whitelist.
CustomerIDOnLocalBlacklist	The customer ID for this transaction is on Vantiv's ThreatMetrix-hosted blacklist.
CustomerIDOnLocalWhitelist	The customer ID for this transaction is on Vantiv's ThreatMetrix-hosted whitelist.
CustomerNameOnLocalBlacklist	The customer name for this transaction is on Vantiv's ThreatMetrix-hosted blacklist.
CustomerNameOnLocalWhitelist	The customer name for this transaction is on Vantiv's ThreatMetrix-hosted whitelist.
DeviceCountriesNotAllowed	This transaction originated from an IP address located in a country on ThreatMetrix-hosted blacklist.
DeviceIDOnThreatMetrixGlobalBlacklist	The originating device is on the ThreatMetrix global blacklist.
DeviceGlobalAgeLessThanOneHour	The originating device was first seen across the entire ThreatMetrix global network within the past hour. This is uncommon and may point to a fraudster simulating a new device through advanced techniques in an attempt to avoid detection.
DeviceLocalAgeLessThanOneHour	The originating device was first seen by Vantiv within the past hour. This may point to a fraudster simulating a new device through advanced techniques in an attempt to avoid detection.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
DeviceNotFingerprinted	ThreatMetrix could not fingerprint the originating device. This is atypical and may indicate a deliberate attempt by the user to cloak his or her identity.
DeviceOnLocalBlacklist	The originating device is on ThreatMetrix-hosted blacklist.
DeviceOnLocalWhitelist	The originating device is on ThreatMetrix-hosted whitelist.
DeviceOnThreatMetrixGlobalBlacklist	The originating device is on the ThreatMetrix global blacklist.
DeviceRejectedByNetwork10Times	The originating device has been rejected by one of ThreatMetrix's customers and/or partners 10 or more times on the suspicion of fraud.
DeviceRejectedByNetwork25Times	The originating device has been rejected by one of ThreatMetrix's customers and/or partners 25 or more times on the suspicion of fraud.
DeviceRejectedByNetwork5Times	The originating device has been rejected by one of ThreatMetrix's customers and/or partners 5 or more times on the suspicion of fraud.
DeviceRejectedByNetworkInLastWeek	The originating device has been rejected by one of ThreatMetrix's customers and/or partners in the last week on the suspicion of fraud.
DeviceReviewedByNetwork5Times	The originating device has been reviewed by one of ThreatMetrix's customers and/or partners 5 or more times on the suspicion of fraud.
DeviceReviewedByNetwork10Times	The originating device has been reviewed by one of ThreatMetrix's customers and/or partners 10 or more times on the suspicion of fraud.
DeviceReviewedByNetwork25Times	The originating device has been reviewed by one of ThreatMetrix's customers and/or partners 25 or more times on the suspicion of fraud.
EmailDistanceTraveled	This email address has been associated with transactions originating from locations at least 1,000 miles apart in the last hour. This is a red flag and warrants caution.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
EmailHostnameTooLong	The hostname portion (i.e. to the right of "@") of this email address exceeds 30 characters. Email addresses associated with suspicious domain names are often used as part of attacks. Overly long hostnames are a common marker of such domains.
EmailHostnameWithNonLetters	The hostname portion (i.e. to the right of "@") of this email address contains non-letter characters (e.g. numbers and special characters). Email addresses associated with suspicious domain names are often used as part of attacks. Hostnames with non-letter characters are a common marker of such domains.
EmailOnLocalBlacklist	The customer email address for this transaction is on Vantiv's ThreatMetrix-hosted blacklist.
EmailOnLocalWhitelist	The customer email address for this transaction is on Vantiv's ThreatMetrix-hosted whitelist.
EmailOnThreatMetrixGlobalBlacklist	This email address is on the ThreatMetrix global blacklist.
EmailRejectedByNetwork10TimesInLastDay	The associated email address has been rejected by one of ThreatMetrix's customers and/or partners 10 or more times in the last day on the suspicion of fraud.
EmailRejectedByNetworkInLastWeek	A transaction using this email address has been rejected by one of ThreatMetrix's customers and/or partners in the last week on the suspicion of fraud.
EmailUsernameTooLong	The name portion (i.e. to the left of "@") of this email address exceeds 30 characters. Email addresses associated with suspicious user names are often used as part of attacks. Overly long user names are a common marker of such domains.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
EmailUsernameWithNonLetters	The name portion (i.e. to the left of "@") of this email address contains non-letter characters (e.g. numbers and special characters). Email addresses associated with suspicious usernames are often used as part of attacks. Usernames with non-letter characters are a common marker of such domains.
ExcessivePaymentsOnDeviceHour	An abnormally high number of transactions have been submitted from this device in the last hour. This is a common indicator of fraudulent payment.
ExcessivePaymentsOnDeviceDay	An abnormally high number of transactions have been submitted from this device in the last 24 hours. This is a common indicator of fraudulent payment.
ExcessivePaymentsOnFuzzyDeviceHour	An abnormally high number of transactions appear to have been submitted from this device in the last hour. This is a common indicator of fraudulent payment.
ExcessivePaymentsOnFuzzyDeviceDay	An abnormally high number of transactions appear to have been submitted from this device in the last 24 hours. This is a common indicator of fraudulent payment.
FlashBrowserLanguageMismatch	The language used by the web browser used to submit this transaction does not match the language used by the Flash plug-in. This is atypical and may be an indicator of misuse.
FlashCookiesJavascriptDisabled	The browser used to submit this transaction has disabled Flash objects, cookies, and JavaScript. This is common to fraud attacks and may be an indicator of misuse.
FlashCookiesDisabled	The browser used to submit this transaction has disabled Flash objects and cookies. This is common to fraud attacks and may be an indicator of misuse.
FlashDisabled	The browser used to submit this transaction has disabled Flash objects. This is common to fraud attacks and may be an indicator of misuse.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
FlashImagesCookiesDisabled	The browser used to submit this transaction has disabled Flash objects, images, and cookies. This is common to fraud attacks and may be an indicator of misuse.
FlashImagesCookiesJavascriptDisabled	The browser used to submit this transaction has disabled Flash objects, images, cookies, and JavaScript. This is common to fraud attacks and may be an indicator of misuse.
FlashImagesDisabled	The browser used to submit this transaction has disabled Flash objects and images. This is common to fraud attacks and may be an indicator of misuse.
FlashImagesJavascriptDisabled	The browser used to submit this transaction has disabled Flash objects, images, and JavaScript. This is common to fraud attacks and may be an indicator of misuse.
FlashJavascriptDisabled	The browser used to submit this transaction has disabled Flash objects and JavaScript. This is common to fraud attacks and may be an indicator of misuse.
FuzzyDeviceLocalAgeLessThanOneHour	The originating device appears to have been seen by Vantiv for the first time within the past hour. This may point to a fraudster simulating a new device through advanced techniques in an attempt to avoid detection.
FuzzyDeviceOnLocalBlacklist	The originating device appears to be on Vantiv's ThreatMetrix-hosted blacklist.
FuzzyDeviceOnLocalWhitelist	The originating device appears to be on Vantiv's ThreatMetrix-hosted whitelist.
FuzzyDeviceOnThreatMetrixGlobalBlacklist	The originating device appears to be on the ThreatMetrix global blacklist.
FuzzyDeviceRejectedByNetworkInLastWeek	The originating device appears to have been rejected by one of ThreatMetrix's customers and/or partners in the last week on the suspicion of fraud.
GeolocationLanguageMismatch	The language detected from the originating web browser is not appropriate for the location. This is atypical and may be an indicator of misuse.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
HiddenProxy	This transaction was submitted through a hidden web proxy, a method that is sometimes employed when trying to cloak one's identity.
ImagesCookiesDisabled	The browser used to submit this transaction has disabled images and cookies. This is common to fraud attacks and may be an indicator of misuse.
ImagesCookiesJavascriptDisabled	The browser used to submit this transaction has disabled images, cookies, and JavaScript. This is common to fraud attacks and may be an indicator of misuse.
ImagesDisabled	The browser used to submit this transaction has disabled images. This is common to fraud attacks and may be an indicator of misuse.
ImagesJavascriptDisabled	The browser used to submit this transaction has disabled images and JavaScript. This is common to fraud attacks and may be an indicator of misuse.
IPHasNegativeReputation	The originating IP address is a potential threat based upon analysis of its activity across the ThreatMetrix network.
IPOnLocalBlacklist	The originating IP address is on ThreatMetrix-hosted blacklist.
IPOnLocalWhitelist	The originating IP address is on ThreatMetrix-hosted whitelist.
IPOnThreatMetrixGlobalBlacklist	The originating IP address is on the ThreatMetrix global blacklist.
IPRejectedByNetwork10Times	The originating IP address has been rejected by one of ThreatMetrix's customers and/or partners 10 or more times on the suspicion of fraud.
IPRejectedByNetwork25Times	The originating IP address has been rejected by one of ThreatMetrix's customers and/or partners 25 or more times on the suspicion of fraud.
IPRejectedByNetwork5Times	The originating IP address has been rejected by one of ThreatMetrix's customers and/or partners 5 or more times on the suspicion of fraud.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
JavascriptDisabled	The browser used to submit this transaction has disabled JavaScript. This is common to fraud attacks and may be an indicator of misuse.
KnownVPNISP	This transaction was submitted through a known Virtual Private Network (VPN), a method that is sometimes employed when trying to cloak one's identity.
MalwareDetectedOnDevice	The originating device appears have to been infected with malware.
OpenProxy	This transaction was submitted through an open web proxy, a method that is sometimes employed when trying to cloak one's identity.
PaymentCardBINShippingAddressGeolocationMismatch	The customer's ship-to address country does not match that of the payment card's issuing bank. This may be an indicator of a fraud attack.
PaymentCardBINTrueIPGeolocationMismatch	The geolocation of the True IP address does not match that of the payment card's issuing bank. This may be an indicator of a fraud attack.
PaymentCardDistanceTraveled	This payment card has been associated with transactions originating from locations at least 1,000 miles apart in the last hour. This is a red flag and warrants caution.
PaymentCardOnThreatMetrixGlobalBlacklist	This payment card is on the ThreatMetrix global blacklist.
PaymentCardRejectedByNetworkInLastWeek	A transaction using this payment card has been rejected by one of ThreatMetrix's customers and/or partners in the last week on the suspicion of fraud.
PhoneNumberOnLocalBlacklist	The customer telephone number for this transaction is on Vantiv's ThreatMetrix-hosted blacklist.
PhoneNumberOnLocalWhitelist	The customer telephone number for this transaction is on Vantiv's ThreatMetrix-hosted whitelist.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
PossibleCookieWipingDay	The user appears to have cleared his or her browser's cookies 3 or more times in the last day. This is common to fraud attacks and may be an indicator of misuse.
PossibleCookieWipingHour	The user appears to have cleared his or her browser's cookies 3 or more times in the last hour. This is common to fraud attacks and may be an indicator of misuse.
PossibleCookieWipingWeek	The user appears to have cleared his or her browser's cookies 3 or more times in the last week. This is common to fraud attacks and may be an indicator of misuse.
PossibleVPNOrTunnel	This transaction may have been submitted through a Virtual Private Network (VPN), a method that is sometimes employed when trying to cloak one's identity.
PossibleVPNConnection	This transaction may have been submitted through a Virtual Private Network (VPN), a method that is sometimes employed when trying to cloak one's identity.
PotentialVirtualMachine	This transaction may have been submitted using a Virtual Machine, a method that is sometimes employed when trying to cloak one's identity.
ProxyHasNegativeReputation	The originating IP proxy is a potential threat based upon an analysis of its activity across the ThreatMetrix network.
ProxyIPHasNegativeReputation	The originating IP address is a potential threat based upon analysis of its activity across the ThreatMetrix network.
ProxyIPOnLocalBlacklist	The originating proxy IP address is on Vantiv's ThreatMetrix-hosted blacklist.
ProxyIPOnLocalWhitelist	The originating proxy IP address is on Vantiv's ThreatMetrix-hosted whitelist.
ProxyIPOnThreatMetrixGlobalBlacklist	The originating proxy IP address is on the ThreatMetrix global blacklist.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
SatelliteISP	This transaction was submitted through a Satellite Internet Service Provider, a method that is sometimes employed when trying to cloak one's identity.
SatelliteProxyISP	This transaction was submitted through a Satellite Proxy Internet Service Provider, a method that is sometimes employed when trying to cloak one's identity.
SessionAnomaly	Characteristics of the originating device appear to have been modified during the course of the user's web session. This is atypical and may indicate misuse.
ShippingAddressTrueIPGeolocationMismatch	The shipping address country does not match that of the True IP address. This may be an indicator of a package redirection/interception/forwarding or re-shipping fraudster attack.
SuspectedSessionCloaking	The characteristics of the originating browser are consistent with common fraud attacks, and may be an indicator of a fraudster's deliberate attempt to cloak his or her identity.
SuspectedTORNetwork	This transaction appears to have originated from a TOR network, a common source of fraud attacks.
SystemStateAnomaly	The system state of the originating device has changed two or more times within the past hour. This is atypical and may indicate misuse.
TimeZoneTrueGeolocationMismatch	The time zone setting on the originating device does not match to the true geolocation of the customer. This is atypical and may be an indicator of misuse.
TransparentProxy	This transaction was submitted through a transparent web proxy, a method that is most often used in corporate environments, though also employed when trying to cloak one's identity.
TrueIPDNSGeolocationMismatch	The geolocation of the True IP address does not match that of the DNS provider. This may be an indicator of a fraudster's attempt to cloak his or her identity.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
TrueIPHasNegativeReputation	The originating IP address is a potential threat based upon analysis of its activity across the ThreatMetrix network.
TrueIPOnLocalBlacklist	The originating true IP address is on Vantiv's ThreatMetrix-hosted blacklist.
TrueIPOnLocalWhitelist	The originating true IP address is on Vantiv's ThreatMetrix-hosted whitelist.
TrueIPOnThreatMetrixGlobalBlacklist	The originating true IP address is on the ThreatMetrix global blacklist.
TrueIPProxyIPCityMismatch	The city of the True IP address does not match that of the Proxy IP address. This may be an indicator of a fraudster's attempt to cloak his or her identity.
TrueIPProxyIPGeolocationMismatch	The geolocation of the True IP address does not match that of the Proxy IP address. This may be an indicator of a fraudster's attempt to cloak his or her identity.
TrueIPProxyIPISPMismatch	The Internet Service Provider of the True IP address does not match that of the Proxy IP address. This may be an indicator of a fraudster's attempt to cloak his or her identity.
TrueIPProxyIPOrganizationMismatch	The organization of the True IP address does not match that of the Proxy IP address. This may be an indicator of a fraudster's attempt to cloak his or her identity.
TrueIPProxyIPRegionMismatch	The region of the True IP address does not match that of the Proxy IP address. This may be an indicator of a fraudster's attempt to cloak his or her identity.
TrueIPRejectedByNetwork10TimesInLastDay	The originating IP address has been rejected by one of ThreatMetrix's customers and/or partners 10 or more times in the last day on the suspicion of fraud.
TrueIPRejectedByNetworkInLastWeek	The originating true IP has been rejected by one of ThreatMetrix's customers and/or partners in the last week on the suspicion of fraud.

TABLE A-6 Triggered Rules Definitions

Triggered Rule Name	Description
UnusualProxyAttributes	This web proxy used to submit the transaction has unusual attributes (e.g. dialup), which may indicate an attempt to cloak one's identity.

A.7 XML Validation Error Messages

[Table A-7](#) 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".

For information about response values 2 through 5, please refer to [Additional Response Header Error Messages on page 861](#).

TABLE A-7 Response Header Error Message Examples

Example Message (message attribute of littleOnlineResponse)	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 <expDate> element, which has a minLength 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 <expDate> element, which has a maxLength 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 <type> 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 "amount" is incomplete.	The <amount> element does not contain a valid value. For example, if you submitted a captureGivenAuth request and included the amount element without specifying a value, the system returns this error message.
Error validating xml data against the schema on line 6 tag name "echeckSale" 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 (<echeckSale> in the example), as well as the possible valid elements.

TABLE A-7 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 LittleXML you are using.</p>
Error validating xml data against the schema on line 12786. The entity name must immediately follow the <code>&</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>"</code> or <code>&</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 & 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><“symbol</code> 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 only) The system has determined that the Batch 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.8 Additional Response Header Error Messages

When submitting transactions via Open Access, there are additional HTTP responses and validation errors that may occur. The table below provides information about these responses/error messages, as well as some additional errors you may encounter during Online transaction processing.

NOTE: The response value and message in the table represent the values for the response and message attributes of either a `littleResponse` or `littleOnlineResponse`.
If Resubmit? = No, you must debug and modify the submitted message prior to resubmission in order to have any chance of success. Resubmitting the same message will result in the same failure.

TABLE A-8 HTTP Status Message and Validation Errors

response value	message	HTTP Status Code/Message	Description	Resubmit?
2	Invalid XML. Contact support@little.com.	200 OK	The submission is not valid XML containing the <code>user</code> and <code>password</code> elements.	No - debug or contact Vantiv Support
3	Invalid credentials. Contact support@little.com.	200 OK	The submission contains empty or invalid credentials (<code>user</code> and <code>password</code>).	Maybe - verify credentials
4	Connection limit exceeded. Contact support@little.com.	200 OK	The merchant has exceeded the maximum number of concurrent connections.	Yes
5	Objectionable content detected. Contact support@little.com.	200 OK	The system has determined that the submission may contain objectionable content.	No - debug or contact Vantiv Support
N/A	N/A	403 Method Not Allowed	The merchant has exceeded the maximum number of concurrent connections.	Yes
N/A	N/A	405 Method Not Allowed	Only HTTP POST method is allowed.	No - debug or contact Vantiv Support

TABLE A-8 HTTP Status Message and Validation Errors

response value	message	HTTP Status Code/Message	Description	Resubmit?
N/A	N/A	404 Not Found	An invalid URI was used. Verify the URI you are using is correct and that you have not appended any parameters to the URI.	No - debug or contact Vantiv Support
N/A	N/A	417 Expectation Failed	An HTTP Expect header was included in the HTTP POST, which is not allowed.	No - debug or contact Vantiv Support

A.9 ACH Return Reason Codes

Table A-9 is a list of ACH Return Reason Codes, which can apply to either eCheck transactions, or Dynamic Payout funding instructions. These codes are not returned in the LittleXML response messages, but are visible in iQ on the eCheck Returns Received report, as well as the Payment Detail screen and the Funding Instruction Detail screen.

NOTE: If an eCheck is returned for reason Code R01 or R09, it is eligible for redeposit.

TABLE A-9 eCheck Return Reason Codes

ACH 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

TABLE A-9 eCheck Return Reason Codes

ACH Return Reason Code	Description
R17	Item returned because of invalid data; refer to addenda for information
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

TABLE A-9 eCheck Return Reason Codes

ACH Return Reason Code	Description
R44	Invalid individual ID
R45	Invalid individual name or company name
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 time frame
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 time frame
R73	Timely original return - RDFI certifies the original return entry was sent within established time frame 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

TABLE A-9 eCheck Return Reason Codes

ACH Return Reason Code	Description
R84	Cross-border entry not processed by originating gateway operator

A.10 ACH NoC Change Codes

Table A-10 is a list of ACH NOC Change Codes. These codes are included in the daily NOC report made available to you via sFTP.

TABLE A-10 eCheck NOC Change Codes

ACH NOC Change 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)

A.11 Canadian eCheck Return Codes

Table A-9 is a list of Canadian eCheck Return Reason Codes. These codes are not returned in the LittleXML response messages, but are visible in iQ on the eCheck Returns Received report, as well as the Payment Detail screen and the Funding Instruction Detail screen.

NOTE:	Return Reason Codes R901 or R908 are eligible for redeposit. Similarly, Code R900-xx are Edit rejects, which you can resubmit with corrected information.
--------------	---

TABLE A-11 Canadian eCheck Return Reason Codes

Canadian eCheck Return Reason Code	Description
R900-07	Institution ID Invalid
R900-08	Account Number Invalid
R900-16	Institution ID for Return Invalid
R900-17	Account Number for Return Invalid
R900-D2	Destination Institute is Not Defined on FIF
R900-E1	Destination Account Number Invalid
R900-L2	Institution for Return Not Defined on FIF
R900-L3	Institution for Return Cross Reference Invalid
R900-M1	Account for Return Invalid
R901	NFS (Debit Only)
R902	Cannot Trace
R903	Payment Stopped/Recalled
R904	Post/Stale Dated
R905	Account Closed
R907	No Debit Allowed
R908	Funds Not Cleared
R909	Currency/Account Mismatch
R910	Payor/Payee Deceased
R911	Account Frozen

TABLE A-11 Canadian eCheck Return Reason Codes

Canadian eCheck Return Reason Code	Description
R912	Invalid/Incorrect Account Number
R914	Incorrect Payor/Payee Name
R915	PAD No Agreement Existed - Business/Personal
R916	PAD Not According to Agreement - Personal
R917	PAD Agreement Revoked - Personal
R918	PAD No Confirmation/Pre-Notification - Personal
R919	PAD Not According to Agreement - Business
R920	PAD Agreement Revoked - Business
R921	PAD No Confirmation/Pre-Notification - Business
R922	Customer Initiated Return - CREDIT only
R990	Institution in Default

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 proposes 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 should be submitted as Discover.
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	30000000-30599999 30950000-30959999 35280000-35899999 36 38 39 64 65 6011 62212600-62699999 62400000-62699999 62820000-62889999	14 digits or 16 digits	3 digits	
JCB	35 (except 352800-358999)	16 digits	3 digits	Account numbers 352800-358999 are processed through the Discover network
MasterCard	51-55 222100 - 272099	16 digits or 19 digits	3 digits	The 222100 - 272099 BIN range will be active as of October 2016. Merchants must support this BIN range no later than January 1, 2017.
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 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		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.

TEST CARD NUMBERS

The following table provides a list of credit card numbers you can use in our Certification environment to construct your own transactions beyond what is required for certification. These account numbers are extracted from the Certification tests of Chapter 2 and the transaction examples of Chapter 3. Never use these account numbers in the live, production environment.

IMPORTANT: Per PCI DSS Requirements and Security Assessment Procedure, Section 6.4.3, "Production data (live PANs) are not used for testing or development."

TABLE C-1 Test Card Numbers

Account Number	Card Type	CVV2/CID
4457010000000009	Visa	349
4457010100000008	Visa	992
4457010140000141	Visa	N/A
4457010200000247	Visa	N/A
4100200300011001	Visa	463
4100200300012009	Visa	N/A
4100200300013007	Visa	N/A
4100200310000002	Visa	N/A
4024720001231239	Visa	N/A
4457012400000001	Visa	N/A
4457013200000001	Visa	N/A
4457119922390123	Visa	N/A

TABLE C-1 Test Card Numbers

Account Number	Card Type	CVV2/CID
4457000300000007	Visa	N/A
4457000100000009	Visa	N/A
4457003100000003	Visa	N/A
4457000400000006	Visa	N/A
4457000200000008	Visa	N/A
4457000800000002	Visa	N/A
4457000900000001	Visa	N/A
4457001000000008	Visa	N/A
4005550000081019	Visa	N/A
4000000000000002	Visa	555
2223000148400010	MasterCard	001
222300048400011	MasterCard	001
2223280062080010	MasterCard	001
2222630061560019	MasterCard	001
2222470061880012	MasterCard	001
2222400061240016	MasterCard	001
2222400041240011	MasterCard	001
2223520063560019	MasterCard	001
2223520043560014	MasterCard	001
2222420040560011	MasterCard	001
2222410040360017	MasterCard	001
2223020040760014	MasterCard	001
5112000100000003	MasterCard	N/A
5112002100000009	MasterCard	N/A
5112002200000008	MasterCard	N/A
5112000200000002	MasterCard	N/A
5112000300000001	MasterCard	N/A
5112000400000000	MasterCard	N/A
5112010400000009	MasterCard	N/A

TABLE C-1 Test Card Numbers

Account Number	Card Type	CVV2/CID
5112000600000008	MasterCard	N/A
5112010000000003	MasterCard	261
5112010100000002	MasterCard	251
5112010140000004	MasterCard	N/A
5500000254444445	MasterCard	N/A
5592106621450897	MasterCard	N/A
5590409551104142	MasterCard	N/A
5587755665222179	MasterCard	N/A
5445840176552850	MasterCard	N/A
5390016478904678	MasterCard	N/A
5112010201000109	MasterCard	N/A
5112010202000108	MasterCard	N/A
5194560012341234	MasterCard	N/A
5435101234510196	MasterCard	987
5112000900000005	MasterCard	N/A
6011010000000003	Discover	758
6011010100000002	Discover	184
6011010140000004	Discover	N/A
375000026600004	American Express	N/A
375001000000005	American Express	N/A
375001010000003	American Express	0421
375001014000009	American Express	N/A
341234567890127	American Express	N/A

PAYFAC™ DYNAMIC PAYOUT

This appendix discusses the Vantiv Instruction-Based Dynamic Payout option provided for use by PayFacs (Payment Facilitators).

Topics covered in this document include:

- Advantages of Using Dynamic Payout
- Overview of Dynamic Payout
- Example of Funding Instructions
- Funding Instruction Certification Testing
- SSR Reports
- Tax ID Validation Process

NOTE: Please also refer to the *PayFac Dynamic Payout FAQ* document, which contains answers to numerous topics including payout timing, split platform processing, report availability, and general process items.

D.1 Advantages of Using Dynamic Payout

Dynamic Payout is a solution platform that controls the distribution of funds using flexible, customized instructions defined by the PayFac. The solution provides a closed-loop transaction life cycle from payment to payout. With one connection for payments and payouts, our solution reduces a PayFac's dependency on other vendors, minimizing cost associated with PCI and reducing scope.

Dynamic Payout is dependent upon the PayFac submitting instructions each time a payout is required. A PayFac can fund merchants on a fixed schedule, such as daily, weekly or monthly, or irregular schedule. PayFacs may even choose to delay funding based on contractual or risk related issues.

The Dynamic Payout funding solution has the following capabilities and benefits:

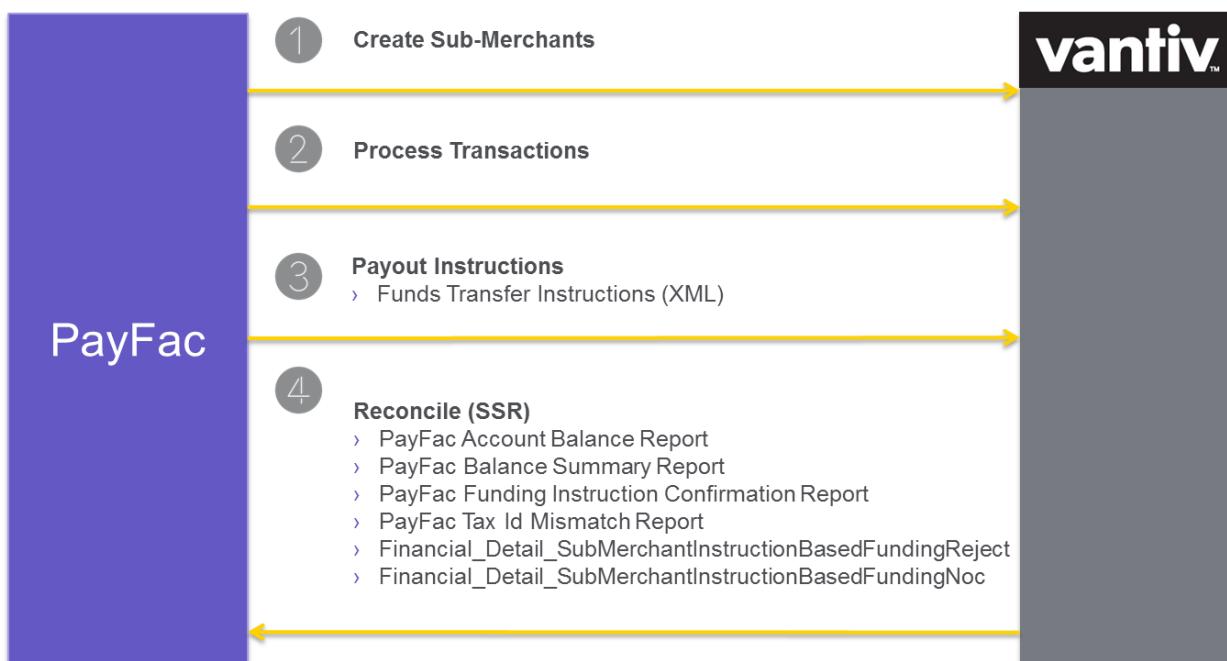
Capabilities:

- Execute payouts for all card brands, including American Express, as early as the next day.
- Determine when to payout.
- Calculate the fees to charge sub-merchants for rendering service. You can use a complex formula or a tiered billing structure.
- Charge fees at a transaction level, not just MID level.
- Maintain a reserve on your sub-merchants.
- Fund a sub-merchant that needs funds split across multiple bank accounts. You can fund multiple bank accounts per sub-merchant.

D.2 Overview of Dynamic Payout

Dynamic Payout is a method of distributing funds to your sub-merchants, physical check (writing) services, other third party vendors (in the payment flow), and yourself via the ACH network. With Dynamic Payout, you board sub-merchants and submit transactions normally. Funds from settled transactions accumulate in a FBO (For Benefit Of) account. You distribute the funds to your sub-merchants by submitting a Batch file containing Funds Transfer Instructions (see [Example of Funding Instructions](#)). You submit these instruction Batch files based upon your payout agreements with your sub-merchants (daily, weekly, monthly, etc.). Vantiv processes the instructions and moves the funds from the holding account to the sub-merchants. Funding can take place as early as the next day after settlement.

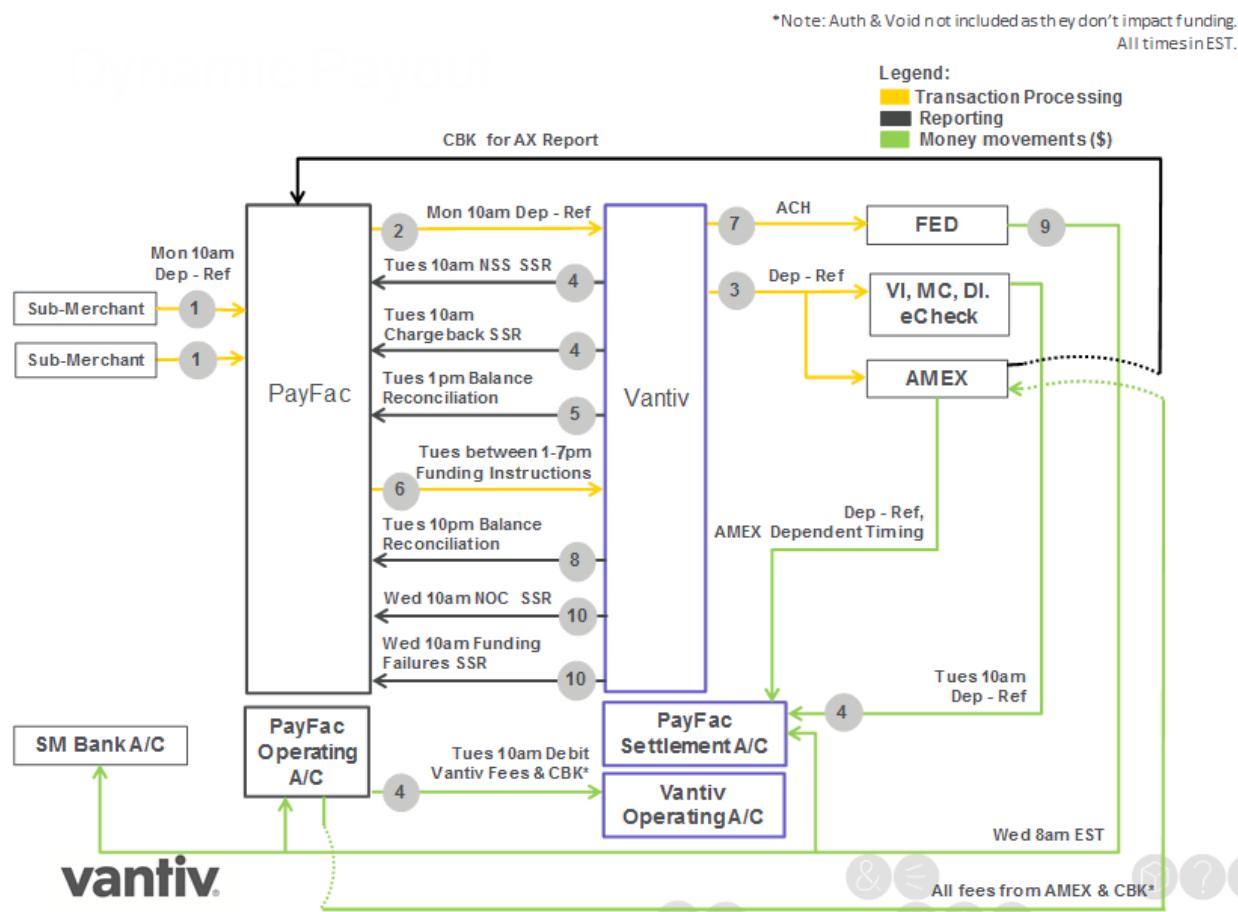
FIGURE D-1 Overview of Instruction-Based Funding



D.2.1 Timing of Transactions, Reports, and Money Movement

The diagram below illustrates the timing and flow of transactions, reports, and money movement. In the diagram, the process begins with the delivery of transactions on Monday morning from the Sub-merchants to the PayFac and completes on Wednesday morning with the final money movement.

FIGURE D-2 Transaction, Reports, and Money Movement



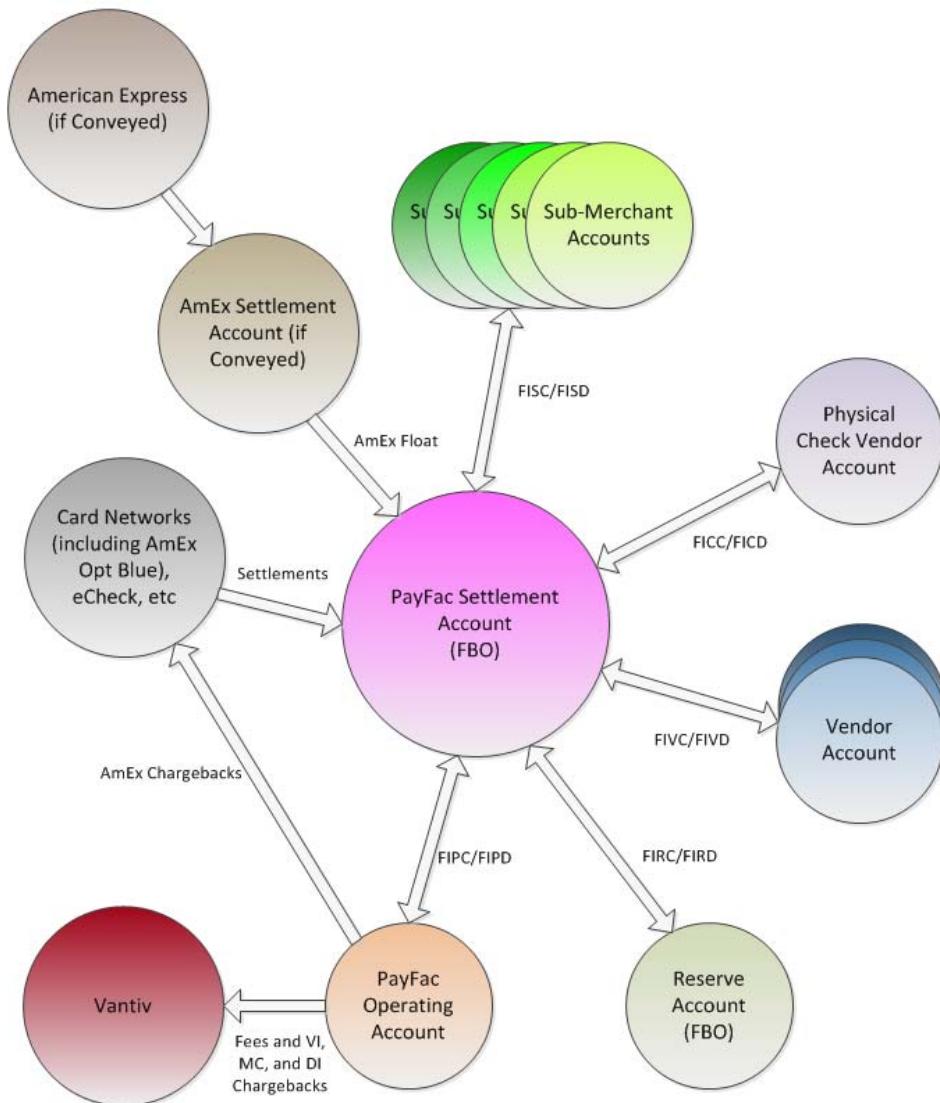
NOTE: The money movement into the PayFac Settlement account from card and eCheck transactions is the Net Settled Sales (i.e., Deposits - Refunds). The funds debited from the PayFac Operating account is the total of Interchange + Chargebacks + Assessments + Vantiv Fees for PayFacs processing on the eComm platform or just Vantiv fees, if processing on the Vantiv Core platform.

D.2.2 Money Movement and Accounts

Figure D-3 below illustrates the various possible accounts, as well as the funding instructions associated with moving the funds between the accounts.

IMPORTANT: Vantiv performs a check on the net (deposits - refunds) money movement between accounts for each Batch of funding instructions. If there are insufficient funds in any account impacted by the funding instructions, Vantiv rejects the entire Batch without processing any instructions. The returned error message provides information about which account lacks the necessary funds.

FIGURE D-3 Dynamic Payout Accounts



The LittleXML file shown in [Example of Funding Instructions](#) on page 885 provides examples of the transactions used for each type of money movement. The following transaction type are available for your use:

- **Funding Instruction PayFac Credit (FIPC)** - used to move funds from the PayFac Settlement account to the PayFac Operating account.
- **Funding Instruction PayFac Debit (FIPD)** - used to move funds from the PayFac Operating account to the PayFac Settlement account.
- **Funding Instruction Reserve Credit (FIRC)** - used to move funds from the PayFac Settlement account to the PayFac Reserve account.
- **Funding Instruction Reserve Debit (FIRD)** - used to move funds from the PayFac Reserve account to the PayFac Settlement account.
- **Funding Instruction Sub-merchant Credit (FISC)** - used to move funds from the PayFac Settlement account to the sub-merchant Operating account.
- **Funding Instruction Sub-merchant Debit (FISD)** - used to move funds from the sub-merchant Operating account to the PayFac Settlement account.
- **Funding Instruction Vendor Credit (FIVC)** - used to move funds from the PayFac Settlement account to the Vendor account.
- **Funding Instruction Vendor Debit (FIVD)** - used to move funds from the Vendor account to the PayFac Settlement account.
- **Funding Instruction Physical Check Credit (FICC)** - used to move funds from the PayFac Settlement account to the Physical Check account.
- **Funding Instruction Physical Check Debit (FICD)** - used to move funds from the Physical Check account to the PayFac Settlement.

D.2.2.1 Account Balance Verifications

Vantiv performs a front-end check on each Funding Instruction Batch, verifying the account balances are adequate to cover the net money movement from each account. If there are insufficient funds in any account impacted by the funding instructions, the entire Batch is rejected. The returned error message provides information about the account lacking funds.

For example, you submit a Batch of funding instructions that include a number of `reserveCredit` and `reserveDebit` transactions, such that the net funds movement (credits - debits) results in \$200,000 being moved from the PayFac Reserve Account to the PayFac Settlement Account. However, the current balance in the Reserve account is only \$175,000. The front-end checks detect this situation and reject the entire Batch with a reject message similar to:

```
<littleResponse version="9.0" xmlns="http://www.little.com/schema" id="691"
  response="1" message="Over Balance (Little ID: 819812345678357001, session
  sequence: 2, unique ID: null) not processed - The specified Funding Instructions
  would result in a negative balance in your Reserve Account"
  littleSessionId="810123456789357102">
</littleResponse>
```

D.3 Example of Funding Instructions

To use Instruction-Based Funding, you must code to LittleXML V9.0 or above. The example below shows a Batch containing the various funding instruction you can use. Do not mix other transaction types in a Batch containing funding instructions.

Example: LittleXML Funding Instructions

```

<littleRequest version="10.2" xmlns="http://www.little.com/schema"
    numBatchRequests="1">
    <authentication>
        <user>username</user>
        <password>password</password>
    </authentication>
    <batchRequest merchantId="01601" numPayFacCredit="1"
        payFacCreditAmount="1000" numPayFacDebit="1" payFacDebitAmount="2000"
        numSubmerchantCredit="1" submerchantCreditAmount="3000"
        numSubmerchantDebit="1" submerchantDebitAmount="4000" numReserveCredit="1"
        reserveCreditAmount="5000" numReserveDebit="1" reserveDebitAmount="6000"
        numVendorCredit="1" vendorCreditAmount="7000" numVendorDebit="1"
        vendorDebitAmount="8000" numPhysicalCheckCredit="1"
        physicalCheckCreditAmount="9000" numPhysicalCheckDebit="1"
        physicalCheckDebitAmount="10000">
        <!-- Example of PayFac funding themselves. Funds move from the PayFac
        Settlement Account to the PayFac Operating Account. -->
        <payFacCredit reportGroup="CollectedFees">
            <!-- ID of Submerchant associated with fee - NOT Payfac ID -->
            <fundingSubmerchantId>SomeSubMerchant</fundingSubmerchantId>
            <!-- Your internal tracking number for fund transfer -->
            <fundsTransferId>123e4567-e89b-12d3-a456-426655440000</fundsTransferId>
            <amount>1000</amount>
        </payFacCredit>
        <!-- Example of PayFac returning money to the settlement account. Funds
        move from the PayFac Operating Account to the PayFac Settlement Account.
        -->
        <payFacDebit reportGroup="MiscRefunds">
            <fundingSubmerchantId>SomeSubMerchant</fundingSubmerchantId>
            <fundsTransferId>123e4567-e89b-12d3-a456-426655440001</fundsTransferId>
            <amount>2000</amount>
        </payFacDebit>
        <!-- Example of PayFac funding the Submerchant. Funds move from the
        PayFac Settlement Account to the Submerchant Account. -->
        <submerchantCredit reportGroup="SubMerchantPayment">
            <fundingSubmerchantId>SomeSubMerchant</fundingSubmerchantId>
            <submerchantName>Some Merchant Inc.</submerchantName>
            <fundsTransferId>123e4567-e89b-12d3-a456-426655440002</fundsTransferId>

```

```

<amount>3000</amount>
<accountInfo>
  <accType>Checking</accType>
  <accNum>123456789012</accNum>
  <routingNum>114567895</routingNum>
</accountInfo>
<customIdentifier>Co_Descriptor</customIdentifier>
</submerchantCredit>
<!-- Example of PayFac debiting the Submerchant. Funds move from the Submerchant Account to the PayFac Settlement Account. --&gt;
&lt;submerchantDebit reportGroup="SubMerchantRefund"&gt;
  &lt;fundingSubmerchantId&gt;SomeSubMerchant&lt;/fundingSubmerchantId&gt;
  &lt;submerchantName&gt;Some Merchant Inc.&lt;/submerchantName&gt;
  &lt;fundsTransferId&gt;123e4567-e89b-12d3-a456-426655440003&lt;/fundsTransferId&gt;
  &lt;amount&gt;4000&lt;/amount&gt;
  &lt;accountInfo&gt;
    &lt;accType&gt;Checking&lt;/accType&gt;
    &lt;accNum&gt;123456789012&lt;/accNum&gt;
    &lt;routingNum&gt;114567895&lt;/routingNum&gt;
  &lt;/accountInfo&gt;
  &lt;customIdentifier&gt;Co_Descriptor&lt;/customIdentifier&gt;
&lt;/submerchantDebit&gt;
<!-- Example of PayFac adding money into reserves. Funds move from the PayFac Settlement Account to the Reserve Account. --&gt;
&lt;reserveCredit reportGroup="Reserve"&gt;
  &lt;fundingSubmerchantId&gt;SomeSubMerchant&lt;/fundingSubmerchantId&gt;
  &lt;fundsTransferId&gt;123e4567-e89b-12d3-a456-426655440004&lt;/fundsTransferId&gt;
  &lt;amount&gt;5000&lt;/amount&gt;
&lt;/reserveCredit&gt;
<!-- Example of PayFac getting money from Reserves. Funds move from the Reserve Account to the PayFac Settlement Account. --&gt;
&lt;reserveDebit reportGroup="SubMerchantRefund"&gt;
  &lt;fundingSubmerchantId&gt;SomeSubMerchant&lt;/fundingSubmerchantId&gt;
  &lt;fundsTransferId&gt;123e4567-e89b-12d3-a456-426655440005&lt;/fundsTransferId&gt;
  &lt;amount&gt;6000&lt;/amount&gt;
&lt;/reserveDebit&gt;
<!-- Example of PayFac funding the vendor. Funds move from the PayFac Settlement Account to the Vendor Account. --&gt;
&lt;vendorCredit reportGroup="vendorPayment"&gt;
  &lt;fundingSubmerchantId&gt;SomeVendor&lt;/fundingSubmerchantId&gt;
  &lt;vendorName&gt;Some Vendor Inc.&lt;/vendorName&gt;
  &lt;fundsTransferId&gt;123e4567-e89b-12d3-a456-426655440006&lt;/fundsTransferId&gt;
  &lt;amount&gt;7000&lt;/amount&gt;
  &lt;accountInfo&gt;
</pre>

```

```

<accType>Checking</accType>
<accNum>123456789012</accNum>
<routingNum>114567895</routingNum>
</accountInfo>
</vendorCredit>

<!-- Example of PayFac debiting the vendor account. Funds move from the
Vendor Account to the Payfac Settlement Account. -->
<vendorDebit reportGroup="vendorReturn">
  <fundingSubmerchantId>SomeVendor</fundingSubmerchantId>
  <vendorName>Some Vendor Inc.</vendorName>
  <fundsTransferId>123e4567-e89b-12d3-a456-426655440007</fundsTransferId>
  <amount>8000</amount>
  <accountInfo>
    <accType>Checking</accType>
    <accNum>123456789014</accNum>
    <routingNum>114567895</routingNum>
  </accountInfo>
</vendorDebit>

<!-- Example of PayFac funding the Physical Check Account. Funds move from
the PayFac Settlement Account to the Physical Check Account -->
<physicalCheckCredit reportGroup="physicalCheck">
  <fundingSubmerchantId>SomeSubMerchant</fundingSubmerchantId>
  <fundsTransferId>123e4567-e89b-12d3-a456-426655440008</fundsTransferId>
  <amount>9000</amount>
</physicalCheckCredit>

<!-- Example of PayFac debiting the Physical Check account. Funds move from
the Physical Check Account to the PayFac Settlement Account-->
<physicalCheckDebit reportGroup="physicalCheckDebit">
  <fundingSubmerchantId>SomeSubMerchant</fundingSubmerchantId>
  <fundsTransferId>123e4567-e89b-12d3-a456-426655440009</fundsTransferId>
  <amount>10000</amount>
</physicalCheckDebit>
</batchRequest>
</littleRequest>

```

Example: Funding Instruction Response

```

<payFacCreditResponse reportGroup="CollectedFees">
  <littleTxnId>82823972759879805</littleTxnId>
  <fundsTransferId>123e4567-e89b-12d3-a456-426655440000</fundsTransferId>
  <response>000</response>
  <responseTime>2014-01-09T20:28:32</responseTime>

```

```
<message>Approved</message>
</payFacCreditResponse>
```

D.3.1 Funding Instruction Void Transactions

You can use the Funding Instruction Void transaction type to void/remove a designated instruction from a submitted batch of instructions provided the following conditions are met:

- You must be coded to XML V10.1 or above.
- Submit the Funding Instruction Void transaction prior to your cutoff time.
- The funding instruction you wish to void must be in a Funding Instruction Batch submitted the same day as the void. This rule also applies to weekends.

Example: Funding Instruction Void Request

```
<littleRequest version="10.1" xmlns="http://www.little.com/schema"
  numBatchRequests = "1">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
  <batchRequest merchantId="PayFacMerch02" numFundingInstructionVoid="1">
    <fundingInstructionVoid reportGroup="void" id="1">
      <littleTxnId>1234</littleTxnId>
    </fundingInstructionVoid>
  </batchRequest>
</littleRequest>
```

Example: Funding Instruction Void Response

```
<littleResponse version="10.1" xmlns="http://www.little.com/schema"
  response="0" message="Valid Format" littleSessionId="82828656962027535">
  <batchResponse littleBatchId="82828656962027543"
  merchantId="PayFacMerch02">
    <fundingInstructionVoidResponse id="1" reportGroup="void">
      <littleTxnId>82828656962109465</littleTxnId>
      <response>001</response>
      <responseTime>2015-11-02T15:51:54</responseTime>
      <message>Transaction Received</message>
    </fundingInstructionVoidResponse>
  </batchResponse>
</littleResponse>
```

D.4 Funding Instruction Certification Testing

In order to validate of your LittleXML structure for Instruction-Based transaction types, submit two Batches, one containing credit transaction and one containing debit transactions, using the data supplied in [Table D-1](#).

NOTE: All test transactions result in approved responses. They are intended only to allow you to verify that your XML messages will pass validation.

Although the tests separate debit and credit transactions, there are no system requirements to do when constructing your Batches.

For required field without supplied data, submit any value valid for the element.

TABLE D-1 Funding Instruction Test Data

Transaction Type	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
First Batch Containing Credit Transactions:				
payFacCredit	<fundsTransferId> <amount>	00001 1000	<fundsTransferId> <response> <message>	00001 000 Approved
submerchantCredit	<fundsTransferId> <amount>	00003 3000	<fundsTransferId> <response> <message>	00003 000 Approved
reserveCredit	<fundsTransferId> <amount>	00005 5000	<fundsTransferId> <response> <message>	00005 000 Approved
vendorCredit	<fundsTransferId> <amount>	00007 7000	<fundsTransferId> <response> <message>	00007 000 Approved
physicalCheckCredit	<fundsTransferId> <amount>	00009 9000	<fundsTransferId> <response> <message>	00009 000 Approved

TABLE D-1 Funding Instruction Test Data

Transaction Type	Supplied Data Elements		Key Response Elements	
	Element	Value	Element	Value
Second Batch Containing Debit Transactions:				
payFacDebit	<fundsTransferId> <amount>	00002 2000	<fundsTransferId> <response> <message>	00002 000 Approved
submerchantDebit	<fundsTransferId> <amount>	00004 4000	<fundsTransferId> <response> <message>	00004 000 Approved
reserveDebit	<fundsTransferId> <amount>	00006 6000	<fundsTransferId> <response> <message>	00006 000 Approved
vendorDebit	<fundsTransferId> <amount>	00008 8000	<fundsTransferId> <response> <message>	00008 000 Approved
physicalCheckDebit	<fundsTransferId> <amount>	00010 10000	<fundsTransferId> <response> <message>	00010 000 Approved

D.5 SSR Reports

You are required to receive the following Scheduled Secure Reports when using the Dynamic Payout feature:

- **Failed Fund Transfer Report** - contains data about failures to transfers funds to Sub-merchants accounts. The report is produced daily.
- **NoC Report** - contains NoC data detailing changes in Sub-merchants accounts discovered during funds transfer operations. The report is produced daily and removed after 24 hours.
- **Account Balance Report**- contains data about balances in various accounts used by this solution. The report is produced daily.
- **TIN Validation Report** - contains data about Legal Entity Tax Identification Numbers validation failures. The report is produced daily.
- **Funding Instruction Discrepancy Report** - provides data about each funding instruction, between 30 to 60 days old, that does not reconcile with submitted Funds In/Out reports.
- **Activity Discrepancy Report** - provides data reconciling the Finds In/Out report, settlement activity, and Net Sales per Source by activity day.
- **Funding Instruction ACH Summary Report** - provides counts of all funding instruction types, as well as counts of NOCs and fund transfer failures for the previous month
- **Funding Instruction Confirmation Report** - provides data about all settled funding instructions from the previous day.
- **Balance Summary Report** - provides a summary of the balance in the PayFac account for the previous day.

In addition to the required reports listed above, you can use the SSR to track transactional data, chargebacks, and to assist in reconciliation operations. Some of these reports are:

- **Net Settled Sales by Transaction Report** - includes all settled and conveyed transactions (deposits and refunds), including echeck transactions. The report can be scheduled based upon either Activity (post) or Settlement (funds transfer) day.
- **Session Report** - includes all transactions for a particular activity post day and allows reconciliation against submitted transactions.
- **Transaction Summary Report** - includes summarized deposits and refunds (both settled and conveyed) submitted by the merchant and broken down by purchase currency, reporting groups, and payment type for a particular activity post day.
- **Activity Report** - includes summarized financial data for transactions (deposits and refunds) based upon activity post date and broken down by Reporting Group and payment type.
- **Settlement Report** - includes summarized financial data for settled transactions (deposits and refunds) based upon settlement (funds transfer) date and broken down by Reporting Group and payment type.

- **Chargeback Financial Report** - includes detailed information about financial impacting chargeback activities for a given activity (post) or fund transfer (settlement) date.
- **Chargeback Status Report** - provides details of all chargeback activities for a designated activity (post day) date or date range in the case of a monthly report. This report is run daily or monthly.
- **Fee Report** - provides a detailed breakdown of all Vantiv and Passthrough (Interchange) fees associated with transactions for a given activity (post) or fund transfer (settlement) date.

NOTE: For additional information about these and other reports, please refer to the latest version of the *Vantiv Scheduled Secure Reports Reference Guide*.

D.6 Tax ID Validation Process

The Internal Revenue Service requires that “a payment settlement entity (PSE) must file Form 1099-K, *Payment Card and Third Party Network Transactions*, for payments made in settlement of reportable payment transactions for each calendar year.”

Vantiv issues 1099-Ks to e-commerce merchants, PayFac, and sub-merchants who are enabled for Dynamic Payout. Prior to issuing 1099-Ks, it is the responsibility of the PayFac to provide and validate sub-merchant Tax ID information to ensure that the validation process performed by Vantiv through IRS.gov is successful.

If the Tax ID validation fails, it is indicated in the Legal Entity Background Check Results panel of the PayFac Portal. You can also view the **PayFac Tax Id Validation Report**, available through the Scheduled Secure Report (SSR) service, to determine if a Legal Entity Tax ID Validation has failed. This report only applies to legal entities that have a sub-merchant funded through our platform, and is produced daily.

You can re-submit the sub-merchant request when certain Tax ID validation errors occur using the **Edit** button at the top of the View Sub-merchant page of the PayFac Portal to edit and re-submit Tax ID information.

Index

Numerics

3DS response certification testing, 2–139

A

AAVS Response Codes, A–838
accNum, 4–338
Account Updater, 1–14
accountInfo, 4–339
accountInformation, 4–340
accountNumber, 4–341
accountNumberLength, 4–342
accountUpdate, 4–343
accountUpdateFilerequestData, 4–344
accountUpdater, 4–345
accountUpdateResponse, 4–349
accType, 4–350
actionReason, 4–351
activate, 4–352
activateResponse, 4–353
activateReversal, 4–354
activateReversalResponse, 4–355
active, 4–356
addOnCode, 4–357
address response certification testing, 2–133
addressIndicator, 4–358
addressLine1, 4–359
Advanced AVS Response Codes, A–838
advancedAVSResult, 4–360
advancedFraudChecks, 4–361
advancedFraudResults, 4–362
affiliate, 4–363
affluence, 4–364
Affluence Indicator Feature, 1–26
allowPartialAuth, 4–365
American Express
 Authorization lifespan, 3–208
amexAggregatorData, 4–366
amount, 4–367
androidpayResponse, 4–369
applepay, 4–370
applepayResponse, 4–371
applicationData, 4–372
applicationExpirationDate, 4–373

applicationPrimaryAccountNumber, 4–374
approvedAmount, 4–375
authAmount, 4–376
authCode, 4–377
authDate, 4–378
authenticatedByMerchant, 4–379
authentication, 4–380
authenticationResult, 4–381
authenticationTransactionId, 4–382
authenticationValue, 4–383
authInformation, 4–384
Authorization
 lifespan, 3–208
authorization, 4–385
Authorization lifespan
 American Express, 1–74
 Discover, 1–74
 MasterCard, 1–74
 PayPal, 1–75
 Visa, 1–74
Authorization Reversal transactions
 certification testing, 2–114
authorizationResponse, 4–386
authorizationSourcePlatform, 4–387
authReversal, 4–388
authReversalResponse, 4–389
availableBalance, 4–390
AVS Filtering, 1–31
AVS response code testing, 2–132
AVS Response Codes, A–837
avsResult, 4–391

B

balanceInquiry, 4–392
balanceInquiryResponse, 4–393
Basic Fraud Tools
 AVS Filtering, 1–31
 Fraud Velocity Filtering, 1–31
 Prepaid Card Filtering, 1–29
 Prior Chargeback Filtering, 1–30
 Prior Fraud Filtering, 1–31
 Security Code No-match Filtering, 1–30
Basic Fraud tools
 International Card Filtering, 1–30

batchRequest, 4–394
 batchResponse, 4–402
 beginningBalance, 4–404
 billingDate, 4–405
 billMeLaterRequest, 4–406
 billMeLaterresponseData, 4–408
 billToAddress, 4–409
 bin, 4–411
 bmlMerchantId, 4–412
 bmlProductType, 4–413
 bypassVelocityCheck, 4–414

C

campaign, 4–415
 cancelSubscription, 4–416
 cancelSubscriptionResponse, 4–416, 4–417
 capability, 4–418
 capture, 4–419
 captureGivenAuth, 4–421
 captureGivenAuthResponse, 4–422
 captureResponse, 4–423
 card, 4–424
 cardAcceptorTaxId, 4–426
 Cardholder Type Indicator Feature, 1–26
 cardholderAuthentication, 4–427
 cardholderId, 4–428
 cardholderName, 4–429
 cardOrToken, 4–430
 cardProductType, 4–431
 cardSuffix, 4–432
 cardValidationNum, 4–433
 cardValidationResult, 4–434
 cashBackAmount, 4–435
 catLevel, 4–436
 ccdPaymentInformation, 4–437
 certification testing
 3DS responses, 2–139
 address responses, 2–133
 Authorization Reversal transactions, 2–114
 AVS response codes, 2–132
 response codes and messages, 2–136
 volume testing, 2–200
 chargeback, 4–438
 checkNum, 4–439
 city, 4–440
 clinicOtherAmount, 4–441
 code, 4–442

commodityCode, 4–443
 companyName, 4–444
 country, 4–445
 createAddOn, 4–446
 createDiscount, 4–447
 createPlan, 4–448
 createPlanResponse, 4–449
 credit, 4–450
 creditLine, 4–452
 creditLittleTxnId, 4–453
 creditResponse, 4–454
 cryptogram, 4–455
 currencyCode, 4–456
 customAttribute1, 4–457
 customBilling, 4–458
 Customer Insight Features, 1–25
 customerInfo, 4–461
 customerIpAddress, 4–462
 customerReference, 4–463
 customerRegistrationDate, 4–464
 customerType, 4–465
 customerWorkTelephone, 4–466
 customIdentifier, 4–460

D

data, 4–467
 deactivate, 4–468
 deactivateResponse, 4–469
 deactivateReversal, 4–470
 deactivateReversalResponse, 4–471
 debtRepayment, 4–472
 Decline Notice
 eCheck Verification, 1–46
 deleteAddOn, 4–473
 deleteDiscount, 4–474
 deliveryType, 4–475
 dentalAmount, 4–476
 depositReversal, 4–477
 depositReversalResponse, 4–478
 description, 4–479
 descriptor, 4–480
 destinationCountryCode, 4–481
 destinationPostalCode, 4–482
 detailTax, 4–483
 deviceManufacturerIdentifier, 4–484
 deviceReputationScore, 4–485
 deviceReviewStatus, 4–486

discountAmount, 4–487
discountCode, 4–488
Discover
 Authorization lifespan, 3–208
dob, 4–489
Duplicate Transaction Detection Feature, 1–7
dutyAmount, 4–490

E

echeck, 4–491
eCheck Auto Redeposit Feature, 1–47
eCheck NoC Update Feature, 1–47
eCheck Processing Feature, 1–45
eCheck Validation Feature, 1–45
eCheck Verification
 Decline Notice, 1–46
eCheck Verification Feature, 1–45
eCheckAccountSuffix, 4–492
echeckCredit, 4–493
echeckCreditResponse, 4–494
echeckForToken, 4–495
echeckOrEcheckToken, 4–496
echeckPreNoteCredit, 4–497
echeckPreNoteCreditResponse, 4–498
echeckPreNoteSale, 4–499
echeckPreNoteSaleResponse, 4–500
echeckRedeposit, 4–501
echeckRedepositResponse, 4–502
echeckSale, 4–503
echeckSalesResponse, 4–504
echeckToken, 4–505
echeckVerification, 4–506
echeckVerificationResponse, 4–507
echeckVoid, 4–508
echeckVoidResponse, 4–509
eciIndicator, 4–510
email, 4–511
employerName, 4–512
encryptedTrack, 4–513
endDate, 4–514
endingBalance, 4–515
endpoint, 4–516
enhancedAuthResponse, 4–517
enhancedData, 4–519
entryMode, 4–523
ephemeralPublicKey, 4–524
error messages

response Header, A–861
XML validation, A–859
expDate, 4–525
expMonth, 4–526
expYear, 4–527
extendedCardResponse, 4–528

F

Features
 Account Updater, 1–14
 Affluence Indicator, 1–26
 Cardholder Type Indicator, 1–26
 Customer Insight, 1–25
 Duplicate Transaction Detection, 1–7
 eCheck Auto Redeposit, 1–47
 eCheck NoC Update, 1–47
 eCheck Processing, 1–45
 eCheck Validation, 1–45
 eCheck Verification, 1–45
 Issuer Country Indicator, 1–26
 Prepaid Indicator, 1–25
 Report Groups, 1–10
 Tokenization, 1–39
fieldValue, 4–529
filtering, 4–530
Filtering Rules, 1–33
finalPayment, 4–531
firstName, 4–532
forceCapture, 4–533
forceCaptureResponse, 4–534
formatId, 4–535
Fraud Check transaction, 1–38
Fraud Filtering
 Filtering Rules, 1–33
Fraud Toolkit, 1–28
Fraud Velocity Filtering, 1–31
fraudCheck, 4–536, 4–712
fraudCheckResponse, 4–537
fraudFilterOverride, 4–538
fraudResult, 4–539
fundingInstructionVoid, 4–540
fundingInstructionVoidResponse, 4–541
fundingSource, 4–542
fundingSubmerchantId, 4–543
fundsTransferId, 4–544

G

giftCardBin, 4–547
giftCardResponse, 4–552

H

header, 4–555
healthcareIIAS, 4–557

I

iban, 4–558
iDEAL, 1–55
idle time timeout, 2–99
IIASFlag, 4–561
incomeAmount, 4–562
incomeCurrency, 4–563
Instruction-Based Dynamic Payout, 1–85
international, 4–565
International Alternate Payment Methods
 iDEAL, 1–55
 SEPA, 1–49
International Card Filtering, 1–30
intervalType, 4–566
invoiceReferenceNumber, 4–567
Issuer Country Indicator Feature, 1–26
issuerCountry, 4–568
itemCategoryCode, 4–569
itemDescription, 4–570
itemDiscountAmount, 4–571
itemSequenceNumber, 4–572

K

ksn, 4–573

L

lastName, 4–574
lineItemData, 4–575
lineItemTotal, 4–577
lineItemTotalWithTax, 4–578
littleInternalRecurringRequest, 4–579
littleOnlineRequest, 4–580
littleOnlineResponse, 4–581
littleRequest, 4–583
littleResponse, 4–584
littleSessionId, 4–586

littleToken, 4–587
littleTxnId, 4–588
load, 4–590
loadResponse, 4–591
loadReversal, 4–592
loadReversalResponse, 4–593

M

mandateProvider, 4–594
mandateReference, 4–595
mandateSignatureDate, 4–596
mandateUrl, 4–597
MasterCard
 Authorization lifespan, 3–208
merchantData, 4–599
merchantGroupId, 4–600
merchantId, 4–601
message, 4–602
middleInitial, 4–603
mpos, 4–604

N

name, 4–605
networkField, 4–606
networkResponse, 4–608
networkSubField, 4–609
networkTransactionId, 4–610
newAccountInfo, 4–611
newCardInfo, 4–612
newCardTokenInfo, 4–613
newTokenInfo, 4–614
nextRecycleTime, 4–615
number, 4–616
numberOfPayments, 4–617

O

onlinePaymentCryptogram, 4–618
optional certification tests
 3DS responses, 2–139
 address responses, 2–133
 AVS response codes, 2–132
 response codes and messages, 2–136
 volume testing, 2–200
orderDate, 4–619
orderId, 4–620
orderSource, 4–621

origAccountNumber, 4–624
 origActionType, 4–625
 origId, 4–627
 originalAccountInfo, 4–623
 originalCard, 4–629
 originalCardInfo, 4–630
 originalCardTokenInfo, 4–631
 originalNetworkTransactionId, 4–632
 originalToken, 4–636
 originalTokenInfo, 4–637
 originalTransactionAmount, 4–638
 origLittleTxnId, 4–640
 origOrderId, 4–641

P

password, 4–642
 payerId, 4–643
 payFacCredit, 4–644
 payFacCreditResponse, 4–645
 payFacDebit, 4–646
 payFacDebitResponse, 4–647
 paymentDataType, 4–648
 paypage, 4–650
 paypageRegistrationId, 4–651
 PayPal
 Authorization lifespan, 3–208
 paypal, 4–652
 payPalNotes, 4–653
 payPalOrderComplete, 4–654
 persistent connection timeout, 2–99
 phone, 4–655
 physicalCheckCredit, 4–656
 physicalCheckCreditResponse, 4–657
 physicalCheckDebit, 4–658
 physicalCheckDebitResponse, 4–659
 pin, 4–660
 planCode, 4–661
 pos, 4–662
 postDate, 4–663
 postDay, 4–664
 preapprovalNumber, 4–665
 preferredLanguage, 4–666
 prepaid, 4–667
 Prepaid Card Filtering, 1–29
 Prepaid Indicator Feature, 1–25
 prepaidCardType, 4–668
 Prior Chargeback Filtering, 1–30

Prior Fraud Filtering, 1–31
 processingInstructions, 4–669
 processingType, 4–670
 productCode, 4–671
 publicKeyHash, 4–672

Q

quantity, 4–673
 Query Transaction, 1–83, 3–300
 queryTransaction, 4–674
 queryTransactionResponse, 4–675
 queryTransactionUnavailableResponse, 4–676

R

recurringRequest, 4–677
 recurringResponse, 4–677, 4–678
 recurringTxnId, 4–679
 recycleAdvice, 4–680
 recycleAdviceEnd, 4–681
 recycleBy, 4–682
 recycleEngineActive, 4–683
 recycleId, 4–684
 recycling, 4–685
 Recycling Engine, 1–12
 Recycling Engine and Auth Reversal, 1–77, 1–85
 recyclingRequest, 4–687
 redirectToken, 4–688
 redirectUrl, 4–689
 refundReversal, 4–691
 refundReversalResponse, 4–692
 registerTokenRequest, 4–693
 registerTokenResponse, 4–694
 reloadable, 4–695
 Report Group Feature, 1–10
 reserveCredit, 4–696
 reserveCreditResponse, 4–697
 reserveDebit, 4–698
 reserveDebitResponse, 4–699
 residenceStatus, 4–700
 response, 4–701
 response codes and messages testing, 2–136
 Response Header error messages, A–861
 responseCode, 4–702
 responseMessage, 4–703
 responseTime, 4–704
 results_Max10, 4–705

RFRRequest, 4–708
 RFRResponse, 4–709
 routingNum, 4–710
 RxAmount, 4–711

S

sale, 4–712
 saleResponse, 4–713
 salesTax, 4–714
 secondaryAmount, 4–715
 Security Code No-match Filtering, 1–30
 sellerId, 4–716
 sellerMerchantCategoryCode, 4–717
 SEPA, 1–49
 sepaDirectDebit, 4–718
 sepaDirectDebitResponse, 4–719
 sequenceType, 4–721
 shipFromPostalCode, 4–722
 shippingAmount, 4–723
 shipToAddress, 4–724
 signature, 4–725
 ssn, 4–728
 startDate, 4–729
 state, 4–730
 Status Query Transaction, 1–83, 3–300
 stopping auto-recycling, 1–77, 1–85
 submerchantCredit, 4–731
 submerchantCreditResponse, 4–732
 submerchantDebit, 4–733
 submerchantDebitResponse, 4–734
 submerchantName, 4–735
 subscription, 4–736
 subscriptionId, 4–738
 surchargeAmount, 4–739

T

Tagging Transactions, 1–11
 taxAmount, 4–741
 taxExempt, 4–742
 taxIncludedInTotal, 4–743
 taxRate, 4–744
 taxType, 4–745
 taxTypeIdentifier, 4–746
 terminalId, 4–747
 termsAndConditions, 4–748
 testing

Authorization Reversal transactions, 2–114
 optional certification tests
 3DS responses, 2–139
 address responses, 2–133
 AVS response codes, 2–132
 response codes and messages, 2–136
 volume testing, 2–200
 threatMetrixSessionId, 4–749
 timeout
 persistent connection, 2–99
 Timeout settings, 2–99
 token, 4–750
 Tokenization Feature, 1–39
 tokenMessage, 4–751
 tokenResponse, 4–752
 tokenResponseCode, 4–753
 totalHealthcareAmount, 4–754
 track, 4–755
 track1Status, 4–756
 track2Status, 4–757
 transactionAmount, 4–758
 transactionId, 4–759
 trialIntervalType, 4–761
 trialNumberOfIntervals, 4–762
 triggeredRule, 4–763
 type, 4–765

U

unitCost, 4–767
 unitOfMeasure, 4–768
 unload, 4–769
 unloadResponse, 4–770
 unloadReversal, 4–771
 unloadReversalResponse, 4–772
 updateAddOn, 4–773
 updateCardValidationNumOnToken, 4–775
 updateCardValidationNumOnTokenResponse, 4–776
 updatedCard, 4–774
 updateDiscount, 4–777
 updatedToken, 4–785
 updatePlan, 4–778
 updatePlanResponse, 4–779
 updateSubscription, 4–780
 updateSubscriptionResponse, 4–784
 url, 4–786
 user, 4–787

V

vendorCredit, 4–788
 vendorCreditResponse, 4–789
 vendorDebit, 4–790
 vendorDebitResponse, 4–791
 vendorName, 4–792
 verificationCode, 4–793
 verify, 4–794
 version, 4–795
 virtualAccountNumber, 4–796
 virtualAuthenticationKeyData, 4–797
 virtualAuthenticationKeyPresenceIndicator, 4–798
 virtualGiftCard, 4–799
 virtualGiftCardBin, 4–800
 virtualGiftCardResponse, 4–801
 Visa
 Authorization lifespan, 3–208
 visionAmount, 4–802
 void, 4–803
 voidResponse, 4–804
 volume certification testing, 2–200

W

wallet, 4–805
 walletSourceType, 4–806
 walletSourceTypeId, 4–807

X

XML elements
 accNum, 4–338
 accountInfo, 4–339
 accountInformation, 4–340
 accountNumber, 4–341
 accountNumberLength, 4–342
 accountUpdate, 4–343
 accountUpdateFileRequestData, 4–344
 accountUpdater, 4–345
 accountUpdateResponse, 4–349
 accType, 4–350
 actionReason, 4–351
 activate, 4–352
 activateResponse, 4–353
 activateReversal, 4–354
 activateReversalResponse, 4–355
 active, 4–356

addOnCode, 4–357
 addressIndicator, 4–358
 addressLine1, 4–359
 advancedAVSResult, 4–360
 advancedFraudChecks, 4–361
 advancedFraudResults, 4–362
 affiliate, 4–363
 affluence, 4–364
 allowPartialAuth, 4–365
 amexAggregatorData, 4–366
 amount, 4–367
 androidpayResponse, 4–369
 applepay, 4–370
 applepayResponse, 4–371
 applicationData, 4–372
 applicationExpirationDate, 4–373
 applicationPrimaryAccountNumber, 4–374
 approvedAmount, 4–375
 authAmount, 4–376
 authCode, 4–377
 authDate, 4–378
 authenticatedByMerchant, 4–379
 authentication, 4–380
 authenticationResult, 4–381
 authenticationTransactionId, 4–382
 authenticationValue, 4–383
 authInformation, 4–384
 authorization, 4–385
 authorizationResponse, 4–386
 authorizationSourcePlatform, 4–387
 authReversal, 4–388
 authReversalResponse, 4–389
 availableBalance, 4–390
 avsResult, 4–391
 balanceInquiry, 4–392
 balanceInquiryResponse, 4–393
 batchRequest, 4–394
 batchResponse, 4–402
 beginningBalance, 4–404
 billingDate, 4–405
 billMeLaterRequest, 4–406
 billMeLaterresponseData, 4–408
 billToAddress, 4–409
 bin, 4–411
 bmlMerchantId, 4–412
 bmlProductType, 4–413
 bypassVelocityCheck, 4–414

campaign, 4-415
cancelSubscription, 4-416
cancelSubscriptionResponse, 4-416, 4-417
capability, 4-418
capture, 4-419
captureGivenAuth, 4-421
captureGivenAuthResponse, 4-422
captureResponse, 4-423
card, 4-424
cardAcceptorTaxId, 4-426
cardholderAuthentication, 4-427
cardholderId, 4-428
cardholderName, 4-429
cardOrToken, 4-430
cardProductType, 4-431
cardSuffix, 4-432
cardValidationNum, 4-433
cardValidationResult, 4-434
cashBackAmount, 4-435
catLevel, 4-436
ccdPaymentInformation, 4-437
chargeback, 4-438
checkNum, 4-439
city, 4-440
clinicOtherAmount, 4-441
code, 4-442
commodityCode, 4-443
companyName, 4-444
country, 4-445
createAddOn, 4-446
createDiscount, 4-447
createPlan, 4-448
createPlanResponse, 4-449
credit, 4-450
creditLine, 4-452
creditLittleTxnId, 4-453
creditResponse, 4-454
cryptogram, 4-455
currencyCode, 4-456
customAttribute1, 4-457
customBilling, 4-458
customerInfo, 4-461
customerIpAddress, 4-462
customerReference, 4-463
customerRegistrationDate, 4-464
customerType, 4-465
customerWorkTelephone, 4-466
customIdentifier, 4-460
data, 4-467
deactivate, 4-468
deactivateResponse, 4-469
deactivateReversal, 4-470
deactivateReversalResponse, 4-471
debtRepayment, 4-472
deleteAddOn, 4-473
deleteDiscount, 4-474
deliveryType, 4-475
dentalAmount, 4-476
depositReversal, 4-477
depositReversalResponse, 4-478
description, 4-479
descriptor, 4-480
destinationCountryCode, 4-481
destinationPostalCode, 4-482
detailTax, 4-483
deviceManufacturerIdentifier, 4-484
deviceReputationScore, 4-485
deviceReviewStatus, 4-486
discountAmount, 4-487
discountCode, 4-488
dob, 4-489
dutyAmount, 4-490
echeck, 4-491
eCheckAccountSuffix, 4-492
echeckCredit, 4-493
echeckCreditResponse, 4-494
echeckForToken, 4-495
echeckOrEcheckToken, 4-496
echeckPreNoteCredit, 4-497
echeckPreNoteCreditResponse, 4-498
echeckPreNoteSale, 4-499
echeckPreNoteSaleResponse, 4-500
echeckRedeposit, 4-501
echeckRedepositResponse, 4-502
echeckSale, 4-503
echeckSalesResponse, 4-504
echeckToken, 4-505
echeckVerification, 4-506
echeckVerificationResponse, 4-507
echeckVoid, 4-508
echeckVoidResponse, 4-509
eciIndicator, 4-510
email, 4-511
employerName, 4-512
encryptedTrack, 4-513
endDate, 4-514

endingBalance, 4–515
endpoint, 4–516
enhancedAuthResponse, 4–517
enhancedData, 4–519
entryMode, 4–523
ephemeralPublicKey, 4–524
expDate, 4–525
expMonth, 4–526
expYear, 4–527
extendedCardResponse, 4–528
fieldValue, 4–529
filtering, 4–530
finalPayment, 4–531
firstName, 4–532
forceCapture, 4–533
forceCaptureResponse, 4–534
formatId, 4–535
fraudCheck, 4–536, 4–712
fraudCheckResponse, 4–537
fraudFilterOverride, 4–538
fraudResult, 4–539
fundingInstructionVoid, 4–540
fundingInstructionVoidResponse, 4–541
fundingSource, 4–542
fundingSubmerchantId, 4–543
fundsTransferId, 4–544
giftCardBin, 4–547
giftCardResponse, 4–552
header, 4–555
healthcareIIAS, 4–557
iban, 4–558
IIASFlag, 4–561
incomeAmount, 4–562
incomeCurrency, 4–563
international, 4–565
intervalType, 4–566
invoiceReferenceNumber, 4–567
issuerCountry, 4–568
itemCategoryCode, 4–569
itemDescription, 4–570
itemDiscountAmount, 4–571
itemSequenceNumber, 4–572
ksn, 4–573
lastName, 4–574
lineItemData, 4–575
lineItemTotal, 4–577
lineItemTotalWithTax, 4–578
littleInternalRecurringRequest, 4–579
littleOnlineRequest, 4–580
littleOnlineResponse, 4–581
littleRequest, 4–583
littleResponse, 4–584
littleSessionId, 4–586
littleToken, 4–587
littleTxnId, 4–588
load, 4–590
loadResponse, 4–591
loadReversal, 4–592
loadReversalResponse, 4–593
mandateProvider, 4–594
mandateReference, 4–595
mandateSignatureDate, 4–596
mandateUrl, 4–597
merchantData, 4–599
merchantGroupId, 4–600
merchantId, 4–601
message, 4–602
middleInitial, 4–603
mpos, 4–604
name, 4–605
networkField, 4–606
networkResponse, 4–608
networkSubField, 4–609
networkTransactionId, 4–610
newAccountInfo, 4–611
newCardInfo, 4–612
newCardTokenInfo, 4–613
newTokenInfo, 4–614
nextRecycleTime, 4–615
number, 4–616
numberOfPayments, 4–617
onlinePaymentCryptogram, 4–618
orderDate, 4–619
orderId, 4–620
orderSource, 4–621
origAccountNumber, 4–624
origActionType, 4–625
origId, 4–627
originalAccountInfo, 4–623
originalCard, 4–629
originalCardInfo, 4–630
originalCardTokenInfo, 4–631
originalNetworkTransactionId, 4–632
originalToken, 4–636

originalTokenInfo, 4–637
originalTransactionAmount, 4–638
origLittleTxnId, 4–640
origOrderId, 4–641
password, 4–642
payerId, 4–643
payFacCredit, 4–644
payFacCreditResponse, 4–645
payFacDebit, 4–646
payFacDebitResponse, 4–647
paymentDataType, 4–648
paypage, 4–650
paypageRegistrationId, 4–651
paypal, 4–652
payPalNotes, 4–653
payPalOrderComplete, 4–654
phone, 4–655
physicalCheckCredit, 4–656
physicalCheckCreditResponse, 4–657
physicalCheckDebit, 4–658
physicalCheckDebitResponse, 4–659
pin, 4–660
planCode, 4–661
pos, 4–662
postDate, 4–663
postDay, 4–664
preapprovalNumber, 4–665
preferredLanguage, 4–666
prepaid, 4–667
prepaidCardType, 4–668
processingInstructions, 4–669
processingType, 4–670
productCode, 4–671
publicKeyHash, 4–672
quantity, 4–673
queryTransaction, 4–674
queryTransactionResponse, 4–675
queryTransactionUnavailableResponse, 4–676
recurringRequest, 4–677
recurringResponse, 4–677, 4–678
recurringTxnId, 4–679
recycleAdvice, 4–680
recycleAdviceEnd, 4–681
recycleBy, 4–682
recycleEngineActive, 4–683
recycleId, 4–684
recycling, 4–685
recyclingRequest, 4–687

redirectToken, 4–688
redirectUrl, 4–689
refundReversal, 4–691
refundReversalResponse, 4–692
registerTokenRequest, 4–693
registerTokenResponse, 4–694
reloadable, 4–695
reserveCredit, 4–696
reserveCreditResponse, 4–697
reserveDebit, 4–698
reserveDebitResponse, 4–699
residenceStatus, 4–700
response, 4–701
responseCode, 4–702
responseMessage, 4–703
responseTime, 4–704
results_Max10, 4–705
RFRRequest, 4–708
RFRResponse, 4–709
routingNum, 4–710
RxAmount, 4–711
sale, 4–712
saleResponse, 4–713
salesTax, 4–714
secondaryAmount, 4–715
sellerId, 4–716
sellerMerchantCategoryCode, 4–717
sepaDirectDebit, 4–718
sepaDirectDebitResponse, 4–719
sequenceType, 4–721
shipFromPostalCode, 4–722
shippingAmount, 4–723
shipToAddress, 4–724
signature, 4–725
ssn, 4–728
startDate, 4–729
state, 4–730
submerchantCredit, 4–731
submerchantCreditResponse, 4–732
submerchantDebit, 4–733
submerchantDebitResponse, 4–734
submerchantName, 4–735
subscription, 4–736
subscriptionId, 4–738
surchargeAmount, 4–739
taxAmount, 4–741
taxExempt, 4–742
taxIncludedInTotal, 4–743

taxRate, 4–744
taxType, 4–745
taxTypeIdentifier, 4–746
terminalId, 4–747
termsAndConditions, 4–748
threatMetrixSessionId, 4–749
token, 4–750
tokenMessage, 4–751
tokenResponse, 4–752
tokenResponseCode, 4–753
totalHealthcareAmount, 4–754
track, 4–755
track1Status, 4–756
track2Status, 4–757
transactionAmount, 4–758
transactionId, 4–759
trialIntervalType, 4–761
trialNumberOfIntervals, 4–762
triggeredRule, 4–763
type, 4–765
unitCost, 4–767
unitOfMeasure, 4–768
unload, 4–769
unloadResponse, 4–770
unloadReversal, 4–771
unloadReversalResponse, 4–772
updateAddOn, 4–773
updateCardValidationNumOnToken, 4–775
updateCardValidationNumOnTokenResponse, 4–776
updatedCard, 4–774
updateDiscount, 4–777
updatedToken, 4–785
updatePlan, 4–778
updatePlanResponse, 4–779
updateSubscription, 4–780
updateSubscriptionResponse, 4–784
url, 4–786
user, 4–787
vendorCredit, 4–788
vendorCreditResponse, 4–789
vendorDebit, 4–790
vendorDebitResponse, 4–791
vendorName, 4–792
verificationCode, 4–793
verify, 4–794
version, 4–795
virtualAccountNumber, 4–796
virtualAuthenticationKeyData, 4–797

virtualAuthenticationKeyPresenceIndicator, 4–798
virtualGiftCard, 4–799
virtualGiftCardBin, 4–800
virtualGiftCardResponse, 4–801
visionAmount, 4–802
void, 4–803
voidResponse, 4–804
wallet, 4–805
walletSourceType, 4–806
walletSourceTypeId, 4–807
yearsAtEmployer, 4–808
yearsAtResidence, 4–809
zip, 4–810

Y

yearsAtEmployer, 4–808
yearsAtResidence, 4–809

Z

zip, 4–810