PayPage Integration Guide

April 2015

XML Release: 9.4

PayPage API V2.1

Document Version: 4.5

Vantiv PayPage Integration Guide Document Version: 4.5

CONTENTS

About This Guide

Intended Audience vii
Revision History vii
Document Structure x
Documentation Set x
Typographical Conventions xi
Contact Information xii

Chapter 1 Introduction

PayPage Overview 2
How PayPage Works 4
Getting Started with PayPage 6
Migrating From Previous Versions of the PayPage API 6
From PayPage with jQuery 1.4.2 6
From JavaScript Browser API to Vantiv-Hosted iFrame 7
Browser and Mobile Operating System Compatibility 8
jQuery Version 8
Certification and Testing Environments 8
Transitioning from Certification to Production 10
Creating a Customized CSS for Vantiv-Hosted iFrame 10
PayPage-Specific Response Codes 11

Chapter 2 Integration and Testing

Integrating Customer Browser JavaScript API Into Your Checkout Page 14
Integration Steps 14
Loading the PayPage API and jQuery 15
Specifying the PayPage API Request Fields 16
Specifying the PayPage API Response Fields 16
Handling the Mouse Click 17
Intercepting the Checkout Form Submission 18
Handling Callbacks for Success, Failure, and Timeout 18
Success Callbacks 18
Failure Callbacks 19
Timeout Callbacks 20
Detecting the Availability of the PayPage API 20
Integrating Vantiv-Hosted iFrame into your Checkout Page 22
Integration Steps 22
Loading the PayFrame 22

Configuring the iFrame 23
Calling the iFrame for the PayPage Registration ID 25
Handling Callbacks 26
Handling Errors 27
Integrating Mobile API Into Your Mobile Application 28
Creating the POST Request 28
Sample Request 29
Sample Response 29
PayPage Support for Apple Pay™ 30
Using the Vantiv Mobile API for Apple Pay 31
Creating a POST Request for an Apple Pay Transaction 33
Sample Apple Pay POST Request 34
Sample Apple Pay POST Response 35
Collecting Diagnostic Information 36
LitleXML Transaction Examples When Using PayPage 37
Transaction Types and Examples 37
Authorization Transactions 38
Authorization Request Structure 38
Authorization Response Structure 39
Sale Transactions 41
Sale Request Structure 41
Sale Response Structure 42
Register Token Transactions 44
Register Token Request 44
Register Token Response 45
Force Capture Transactions 46
Force Capture Request 46
Force Capture Response 47
Capture Given Auth Transactions 48
Capture Given Auth Request 48
Capture Given Auth Response 50
Credit Transactions 51
Credit Request Transaction 51
Credit Response 52
Testing and Certification 54
Testing PayPage Transactions 54

Appendix A Code Samples and Other Information

HTML Checkout Page Examples 60
HTML Example for Non-PayPage Checkout Page 60
HTML Example for JavaScript API-Integrated Checkout Page 61

HTML Example for Hosted iFrame-Integrated Checkout Page 64
Information Sent to Order Processing Systems 68
Information Sent Without Integrating PayPage 68
Information Sent with Browser-Based PayPage Integration 68
Information Sent with Mobile API-Based Integration 69
Sample JavaScripts 70
Sample PayPage JavaScript (litle-api2.js) 70
Sample PayFrame Client (payframe-client.js) 75
Sample PayFrame JavaScript (payframe.js) 78
LitleXML Elements for PayPage 83
cardValidationNum 84
expDate 85
paypage 86
paypageRegistrationId 87
registerTokenRequest 88
registerTokenResponse 89

Appendix B CSS Properties for iFrame API

CSS Property Groups 91
Properties Excluded From White List 105

Appendix C Sample PayPage Integration Checklist

Index

About This Guide

This guide provides information on integrating the PayPage solution, which, when used together with Vault, may help reduce your risk by virtually eliminating your exposure to sensitive cardholder data and help reduce PCI applicable controls. It also explains how to perform PayPage transaction testing and certification with Vantiv.

Intended Audience

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

Revision History

This document has been revised as follows:

TABLE 1 Document Revision History

Doc. Version	Description	Location(s)
1.0	Initial Draft	All
1.1	Second Draft	All
2.0	First full version	All
2.1	Added new material (examples, XML reference information) on submitting a PayPage Registration ID with a Token Request. Also added a new Response Reason Code.	Chapters 1 and 2, and Appendix A
2.2	Changed product name from 'Pay Page' to 'PayPage'.	All
2.3	Changed certification environment URL from https://**merchant1**.securepaypage.litle.com/litle-api.js to https://**cert01**.securepaypage.litle.com/litle-api.js.	All

TABLE 1 Document Revision History

Doc. Version	Description	Location(s)
2.4	Added information for the support of new transaction types (Capture Given Auth, Force Capture, and Credit), including XML Examples, and XML reference information.	Chapter 2 and Appendix A
	Added information and recommendations for timeout periods and failure callbacks. Updated the Getting Started section.	Chapter 1 and 2
2.5	Added additional information on components of the SendtoLitle call and recommendations on collecting data in the case of a failed transaction.	Chapter 2
	Added a new Appendix contained a sample PayPage Integration Checklist.	Appendix B
2.6	Added and updated information due to XML changes in support of CVV2 updates, including coding changes, new test cases, etc.	All chapters and appendixes.
	Updated the sample Litle JavaScript.	Appendix A
	Changed certification environment URL from: https://cert01.securepaypage.litle.com/litle-api.js. to: https://cert01.securepaypage.litle.com/**LitlePayPage**/litle-api.js	Chapter 1 and 2
2.7	Changed the certification and production URLs: **New Testing and Certification URL:** https://request.cert01-securepaypage-litle.com **New Production URL:** (see your Implementation Consultant)	All
2.8	Removed reference to *companyname* in production URL example.	Chapter 2
2.9	Removed references to Production URL.	All
2.10	Added and updated information on the updated Litle API (V2), including requirements on loading a jQuery library.	All
	Added information on migrating from previous versions of the PayPage API.	Chapter 1
	Updated the sample Litle JavaScript.	Appendix A
	Changed certification environment URL from: https://cert01.securepaypage.litle.com/LitlePayPage/litle-api.js to: https://cert01.securepaypage.litle.com/LitlePayPage/litle-api2.js	Chapter 1 and 2

TABLE 1 Document Revision History

Doc. Version	Description	Location(s)
2.11	Added text, notes, and callouts to further emphasize the proper use of the certification environment URL versus the production URL.	All
2.12	Changed the certification environment URL from: https://cert01.securepaypage.litle.com/LitlePayPage/litle-api2.js to: https://request-prelive.np-securepaypage-litle.com/LitlePayPage/litle-api2.js	All
	Added information on the new Certification and Testing environments: Pre-live, Post-live (regression testing), and Sandbox.	Chapter 1
3.0	Removed sections related to alternative processing.	All
3.1	Added information on PayPage capabilities in a native mobile application.	All
4.0	Re-branded the entire guide to reflect Litle-Vantiv merger.	All
	Updated to LitleXML version 8.27.	Chapter 2
4.1	Added information on new fields returned in the PayPage response; updated JavaScript version (2.1).	Chapter 2, Appendix A and B
4.2	Changed the name of the mobile product to native mobile application	All
4.3	Updated verbiage related to PCI scope.	All
	Added information on support for Apple Pay™.	Chapter 2
	Corrected the URL for mobile POST requests.	Chapter 2
4.4	Corrected the URL for in various examples for mobile POST requests from: https://request-prelive.np-securepaypage-litle.com/LitlePayPage to https://request-prelive.np-securepaypage-litle.com/LitlePayPage/**paypage**	Chapter 1 and 2
4.5	Update the guide to include information on the Vantiv-Hosted PayPage iFrame solution (many changes and re-arrangment of sections). Also includes the addition of Appendix B, and new HTML and JavaScript samples in Appendix A.	All

Document Structure

This manual contains the following sections:

Chapter 1, "Introduction"

This chapter provides an overview of the PayPage feature, and the initial steps required to get started with PayPage.

Chapter 2, "Integration and Testing"

This chapter describes the steps required to integrate the PayPage feature as part of your checkout page, LitleXML transaction examples, and information on PayPage Testing and Certification.

Appendix A, "Code Samples and Other Information"

This appendix provides code examples and reference material related to integrating the PayPage feature.

Appendix B, "CSS Properties for iFrame API"

This appendix provides a list of CSS Properties for use with the iFrame implementation of PayPage.

Appendix C, "Sample PayPage Integration Checklist"

This appendix provides a sample of the PayPage Integration Checklist for use during your Implementation process

Documentation Set

The Vantiv documentation set also include the items listed below. Please refer to the appropriate guide for information concerning other Vantiv product offerings.

- *Vantiv iQ Reporting and Analytics User Guide*
- *Vantiv LitleXML Reference Guide*
- *Vantiv eCommerce Solution for Apple Pay*
- *Vantiv Chargeback API Reference Guide*
- *Vantiv Chargeback Process Guide*
- *Vantiv PayPal Integration Guide*
- *Vantiv PayPal Credit Integration Guide*
- *Vantiv PayFac API Reference Guide*
- *Vantiv PayFac Portal User Guide*

- *Vantiv LitleXML Differences Guide*
- *Vantiv Scheduled Secure Reports Reference Guide*
- *Vantiv Chargeback XML and Support Documentation API Reference Guide* **(Legacy)**

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: • 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.
Italicized text	Italic type in text indicates a term defined in the text, the glossary, or in both locations.
blue text	Blue text indicates a hypertext link.

Contact Information

This section provides contact information for organizations within Vantiv

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

Implementation Contact Information

E-mail	implementation@litle.com
Hours Available	Monday – Friday, 8:30 A.M.– 5:30 P.M. EST

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

Technical Support Contact Information

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

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

Customer Experience Management/Customer Service Contact Information

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

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

Chargebacks Department Contact Information

Telephone	978-275-6500 (option 4)
E-mail	chargebacks@litle.com
Hours Available	Monday – Friday, 7:30 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@litle.com

1

INTRODUCTION

This chapter provides an introduction and an overview of the PayPage feature. The topics discussed in this chapter are:

- PayPage Overview
- How PayPage Works
- Getting Started with PayPage
- Migrating From Previous Versions of the PayPage API

NOTE: **The PayPage feature of the Vault Solution operates on JavaScript-enabled browsers only.**

1.1 PayPage Overview

Vantiv' s PayPage solution helps solve your card-not-present challenges by virtually eliminating payment data from your systems. The PayPage solution reduces the threat of account data compromise by transferring the risk to Vantiv, reducing PCI applicable controls. The PayPage feature controls the fields on your checkout page that collect sensitive cardholder data. When the cardholder submits their account information, your checkout page calls PayPage to register the account number for a low-value token, returning a Registration ID--a PCI non-sensitive value--in place of the account number. No card data is actually transmitted via your web server.

Vantiv provides three integration options for PayPage:

- **JavaScript Customer Browser API** - controls the fields on your checkout page that hold sensitive card data. When the cardholder submits his/her account information, your checkout page calls the PayPage JavaScript to register the provided credit card for a token. The JavaScript validates, encrypts, and passes the account number to our system as the first step in the form submission. The return message includes the *PayPage Registration ID* in place of the account number. No card data is actually transmitted via your web server.
- **Vantiv-Hosted iFrame API** - this solution builds on the same architecture of risk- and scope-reducing technologies of PayPage by fully hosting fields with PCI-sensitive values. Payment card fields, such as primary account number (PAN), expiration date, and CVV value, are hosted in our PCI-Compliance environment, rather than embedded as code into your checkout page within your environment.
- **Mobile API -** allows you to use a PayPage-like solution to handle payments without interacting with the PayPage JavaScript in a browser. With Mobile PayPage, you POST an account number to our system and receive a PayPage Registration IDs in response. You can use it in native mobile applications--where the cross-domain limitations of a browser don't apply--in order to achieve a similar reduction in risk as PayPage.

Figure 1-1 and Figure 1-2 illustrate the difference between the Vault and the Vault with PayPage. See the section, How PayPage Works on page 4 for a additional details.

NOTE: In order to optimally use the PayPage feature for the most risk reduction (i.e., to no longer handle primary account numbers), this feature must be used at all times, without exception.

FIGURE 1-1 Vault

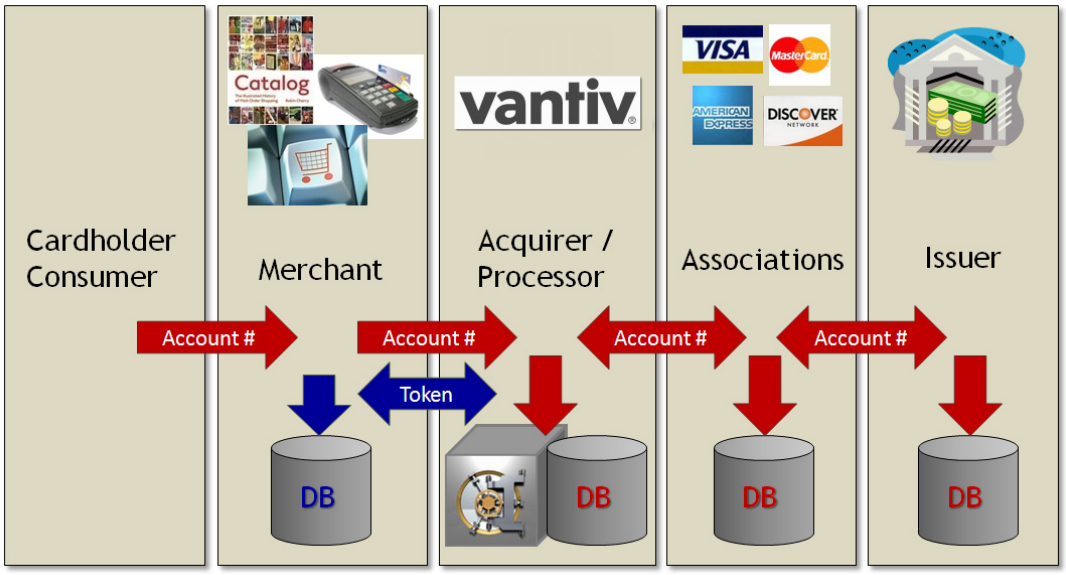
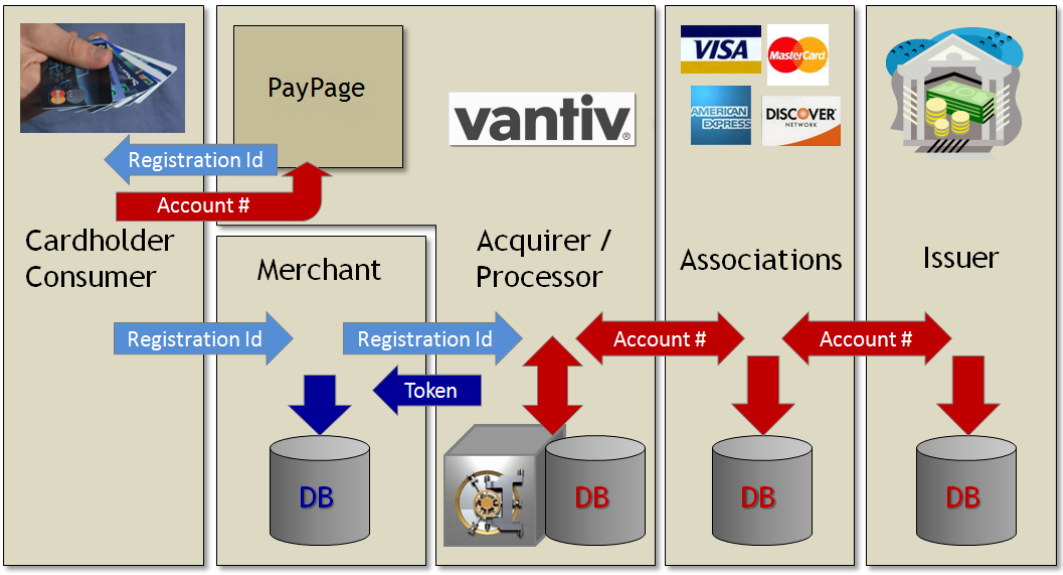
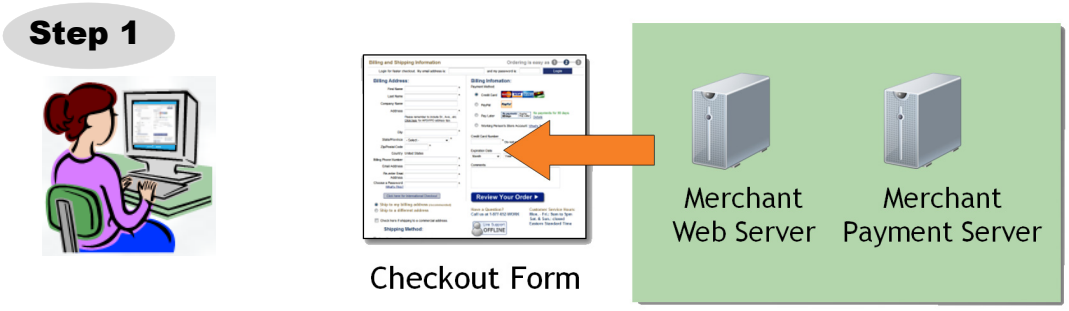


FIGURE 1-2 Vault with PayPage

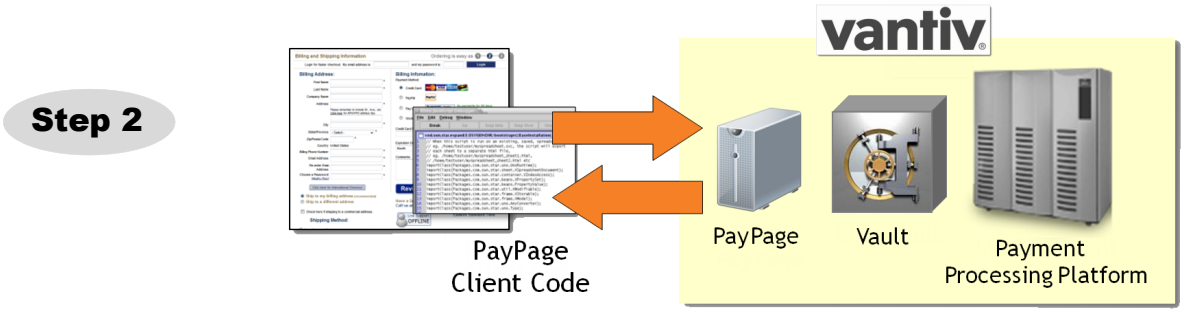


1.2 How PayPage Works

This section illustrates the PayPage process.

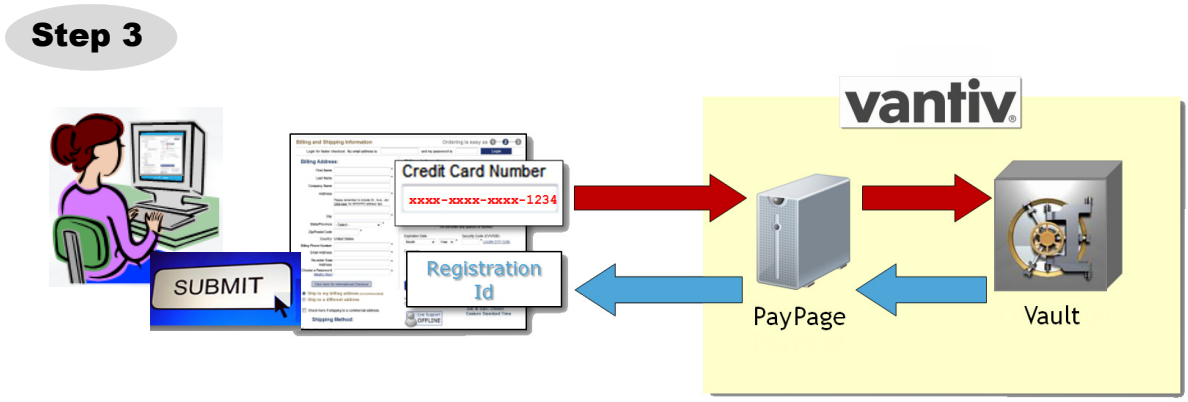


1. When your customer is ready to finalize a purchase from your website or mobile application, your web server delivers your Checkout Form to the customer's web browser or mobile device.

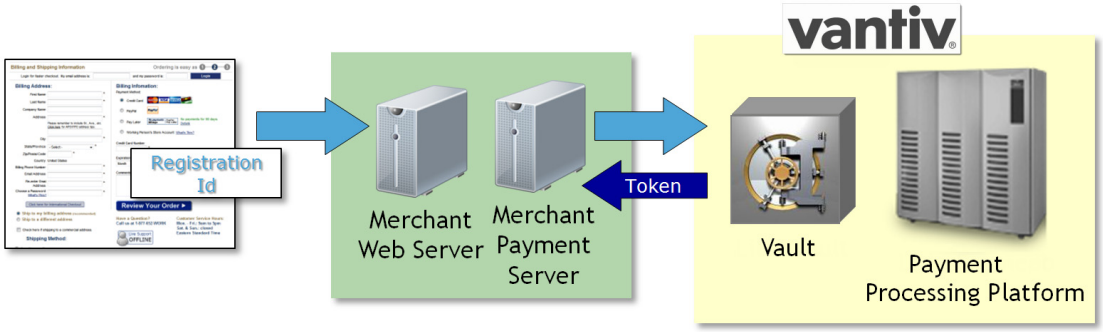


2. The customer's browser loads the PayPage Client code (JavaScript API) from our PayPage server. The API validates credit cards, submits account numbers to the PayPage Service, encrypts account numbers, and adds PayPage registration IDs to the form. It also contains our public key.

(continued on next page)



3. After the customer enters their card number and clicks or taps the submit button, the PayPage API (or POST request) sends the card number data to our PayPage service. The PayPage service submits a LitleXML transaction to the Vault to register a token for the card number provided. The token is securely stored in the Vault for eventual processing (when your payment processing system submits an authorization or sale transaction). A PayPage Registration ID is generated and returned to the customer's browser as a hidden field, or to their mobile device as a POST response.



All of the customer-provided information is then delivered to your web server along with the PayPage Registration ID. Your payment processing system sends the payment with the Registration ID, and the Vault maps the Registration ID to the token and card number, processing the payment as usual. The LitleXML response message contains the token, which you store as you would a credit card.

1.3 Getting Started with PayPage

Before you start using the PayPage feature of the Vault solution, you must complete the following:

- Ensure that your organization is enabled and certified to process tokens, using the Vault solution.
- Complete the PayPage Integration Checklist provided by your Implementation Consultant, and return to Implementation. See Appendix C, "Sample PayPage Integration Checklist".
- Obtain a *PayPage ID* from our Implementation department.
- If you are implementing the Vantiv-Hosted iFrame solution, create a Cascading Style Sheet (CSS) to customize the look and feel of the iFrame to match your checkout page, then submit the Style Sheet to Vantiv for verification. See Creating a Customized CSS for Vantiv-Hosted iFrame on page 10 for more information.
- Modify your checkout page or mobile application--and any other page that receives credit card data from your customers--to integrate the PayPage feature (execute an API call or POST to our system). See one of the following sections, depending on your application:
  - Integrating Customer Browser JavaScript API Into Your Checkout Page on page 14
  - Integrating Vantiv-Hosted iFrame into your Checkout Page on page 22
  - Integrating Mobile API Into Your Mobile Application on page 28 for more information.
- Modify your system to accept the response codes listed in Table 1-2, PayPage-Specific Response Codes Received in Browser or Mobile Device, and Table 1-3, PayPage Response Codes Received in LitleXML Responses.
- Test and certify your checkout process. See Testing and Certification on page 54 for more information.

1.3.1 Migrating From Previous Versions of the PayPage API

1.3.1.1 From PayPage with jQuery 1.4.2

Previous versions of the PayPage API included jQuery 1.4.2 (browser-based use only). Depending on the implementation of your checkout page and your use of other versions of jQuery, this may result in unexpected behavior. This document describes version 2 of the PayPage API, which requires you to use your own version of jQuery, as described within.

If you are migrating, you must:

- Include a script tag to download jQuery before loading the PayPage API.
- Construct a new LitleAPI module when calling `sendToLitle`.

1.3.1.2 From JavaScript Browser API to Vantiv-Hosted iFrame

When migrating from the JavaScript Customer Browser API PayPage solution to the Vantiv-Hosted iFrame solution, complete the following steps. For a full HTML code example a Vantiv-Hosted iFrame PayPage implementation, see the HTML Example for Hosted iFrame-Integrated Checkout Page on page 64.

1. Remove the script that was downloading `litle-api2.js`.
2. Add a script tag to download `payframe-client.min.js`.
3. On your form, remove the inputs for account number, cvv, and expiration date. Put an empty `div` in its place.
4. Consolidate the three callbacks (`submitAfterLitle`, `onErrorAfterLitle` and `onTimeoutAfterLitle` in our examples) into one callback that accepts a single argument. In our example, this is called `payframeClientCallback`.
5. To determine success or failure, inspect `response.response` in your callback. If successful, the response is '870.' Check for time-outs by inspecting the `response.timeout`; if it is defined, a timeout has occurred.
6. In your callback, add code to retrieve the `paypageRegistrationId`, `bin`, `expMonth` and `expYear`. Previously, `paypageRegistrationId` and `bin` were placed directly into your form by our api, and we did not handle `expMonth` and `expYear` (we've included these inside our form to make styling and layout simpler).
7. See Handling Callbacks on page 26 to style the iFrame, and Configuring the iFrame on page 23 to configure the iFrame.
8. See Calling the iFrame for the PayPage Registration ID on page 25 to retrieve the `paypageRegistrationId`.

1.3.2 Browser and Mobile Operating System Compatibility

The PayPage feature is compatible with the following:

Browsers (when JavaScript is enabled):

- Mozilla Firefox 3 and later
- Internet Explorer 8 and later
- Safari 4 and later
- Opera 10.1 and later
- Chrome 1 and later

Native Applications on **Mobile Operating Systems:**

- Chrome Android 40 and later
- Android 2.1 and later
- Apple iOS 3.2 and later
- Windows Phone 10 and later
- Blackberry 7, 10 and later
- Other mobile OS

1.3.3 jQuery Version

If you are implementing a browser-based solution, you must load a jQuery library *before* loading the PayPage API. We recommend using jQuery 1.4.2 or higher. Refer to http://jquery.com for more information on jQuery.

1.3.4 Certification and Testing Environments

For certification and testing of Vantiv feature functionality, including PayPage, Vantiv uses three certification and testing environments:

- **Pre-Live** - this test environment is used for all merchant Certification testing. This environment should be used by both newly on-boarding Vantiv merchants, and existing merchants seeking to incorporate additional features or functionalities (for example, PayPage) into their current integrations.
- **Post-Live** - this test environment is intended for merchants that are already fully certified and submitting transactions to our 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, we migrate merchants that are Production-enabled to the Post-Live environment for any on-going testing needs.

- **Sandbox** - this 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 LitleXML. This is a stateless simulator, so it has no historical transactional data, and does not provide full functionality.

Use the URLs listed in Table 1-1 when testing and submitting PayPage transactions.

TABLE 1-1 PayPage Certification, Testing, and Production URLs

Environment	URL Purpose	URL
Testing and Certification	JavaScript Library	https://request-prelive.np-securepaypage-litle.com/LitlePayPage/litle-api2.js
	Request Submission (excluding POST)	https://request-prelive.np-securepaypage-litle.com
	iFrame	https://request-prelive.np-securepaypage-litle.com/LitlePayPage/payframe-client.min.js
	POST Request Submission (for Mobile API)	https://request-prelive.np-securepaypage-litle.com/LitlePayPage/paypage
Post-Live (Regression Testing)	JavaScript Library	https://request-postlive.np-securepaypage-litle.com/LitlePayPage/litle-api2.js
	Request Submission (excluding POST)	https://request-postlive.np-securepaypage-litle.com
	iFrame	https://request-postlive.np-securepaypage-litle.com/LitlePayPage/payframe-client.min.js
	POST Request Submission (for Mobile API)	https://request-postlive.np-securepaypage-litle.com/LitlePayPage/paypage
Live Production	*Production*	*Contact your Implementation Consultant for the PayPage Production URL.*

1.3.5 Transitioning from Certification to Production

Before using your checkout form with PayPage in a production environment, replace all instances of the Testing and Certification URLs listed in Table 1-1 with the production URL. Contact Implementation for the appropriate production URL. **The URLs in the above table and in the sample scripts throughout this guide should only be used in the certification and testing environment.**

1.3.6 Creating a Customized CSS for Vantiv-Hosted iFrame

Before you begin using the Vantiv-Hosted iFrame solution, you must create a Cascading Style Sheet (CSS) to customize the look and feel of the iFrame to match your checkout page, then submit the style sheet to Vantiv, where it will be tested before it is deployed into production.

Vantiv reviews your CSS by an automatic process which has white-listed allowed CSS properties and black-listed dangerous CSS values (such as URL, JavaScript, expression). Properties identified as such have been removed from the white list, and if used, will fail verification of the CSS. See Table B-23, "CSS Properties Excluded From the White List (not allowed)" for those properties not allowed.

If an error is detected, Vantiv returns the CSS for correction. If the CSS review is successful, the CSS is uploaded to the your PayPage configuration.

Note the following:

- If additional properties and/or values are introduced in future CSS versions, those properties and values will be automatically black-listed until Vantiv can review and supplement the white-listed properties and values.
- Certain properties allow unacceptable values, including URL, JavaScript, or expression. This includes the **content** property, which allows you to enter ‘Exp Date’ instead of our provided ‘Expiration Date’ label. If the property contains a URL, JavaScript, expression, or attr(href), we will fail verification of the CSS.
- Any property in the white list also allows its browser’s extended values, where applicable.

1.3.7 PayPage-Specific Response Codes

Table 1-2 and Table 1-3 list response codes specific to the PayPage feature, received in the browser or mobile device, and those received via a LitleXML Response. For information on response codes specific to token transactions, see the *Vantiv LitleXML Reference Guide*.

TABLE 1-2 PayPage-Specific Response Codes Received in Browser or Mobile Device

Response Code	Description	Error Type	Error Source
870	Success	--	--
871	Account Number not Mod10	Validation	User
872	Account Number too short	Validation	User
873	Account Number too long	Validation	User
874	Account Number not numeric	Validation	User
875	Unable to encrypt field	System	JavaScript
876	Account number invalid	Validation	User
881	Card Validation number not numeric	Validation	User
882	Card Validation number too short	Validation	User
883	Card Validation number too long	Validation	User
889	Failure	System	Litle

TABLE 1-3 PayPage Response Codes Received in LitleXML Responses

Response Code	Response Message	Response Type	Description
877	Invalid PayPage Registration ID	Hard Decline	A PayPage response indicating that the PayPage Registration ID submitted is invalid.
878	Expired PayPage Registration ID	Hard Decline	A PayPage response indicating that the PayPage Registration ID has expired (PayPage Registration IDs expire 24 hours after being issued).
879	Merchant is not authorized for PayPage	Hard Decline	A response indicating that your organization is not enabled for the PayPage feature.

2

INTEGRATION AND TESTING

This chapter describes the steps required to integrate the PayPage feature as part of your checkout form or native mobile application, along with information on PayPage Certification. The topics included are:

- Integrating Customer Browser JavaScript API Into Your Checkout Page
- Integrating Vantiv-Hosted iFrame into your Checkout Page
- Integrating Mobile API Into Your Mobile Application
- Collecting Diagnostic Information
- LitleXML Transaction Examples When Using PayPage
- Testing and Certification

2.1 Integrating Customer Browser JavaScript API Into Your Checkout Page

This section provides step-by-step instructions for integrating the Customer Browser JavaScript API PayPage solution into your checkout page.

See Integrating Mobile API Into Your Mobile Application on page 28 for more information on the mobile solution.

See Integrating Vantiv-Hosted iFrame into your Checkout Page on page 22 for more information on the iFrame solution.

2.1.1 Integration Steps

Integrating PayPage into your checkout page includes these steps, described in detail in the sections to follow:

1. Loading the PayPage API and jQuery
2. Specifying the PayPage API Request Fields
3. Specifying the PayPage API Response Fields
4. Handling the Mouse Click
5. Intercepting the Checkout Form Submission
6. Handling Callbacks for Success, Failure, and Timeout
7. Detecting the Availability of the PayPage API

The above steps make up the components of the `sendToLitle` call:

```
sendToLitle(litleRequest, litleFormFields, successCallback, errorCallback,
timeoutCallback, timeout)
```

- **litleRequest** - captures the form fields that contain the request parameters (`paypageId` , `url`, etc.)
- **litleFormFields** - captures the form fields that we should use to set various portions of the PayPage registration response (PayPage Registration Id, response reason code, response reason message, etc.).
- **successCallback** - specifies the method that we should use to handle a successful PayPage registration.
- **errorCallback** - specifies the method that we should use to handle a failure event (if error code is received).
- **timeoutCallback** - specifies the method that we should use to handle a timeout event (if the `sendToLitle` exceeds the timeout threshold).
- **timeout** - specifies the number of milliseconds before the `timeoutCallback` is invoked.

JavaScript code examples are included with each step. For a full HTML code example of the PayPage implementation, see the HTML Checkout Page Examples on page 60.

2.1.2 Loading the PayPage API and jQuery

To load the PayPage client JavaScript library from the PayPage application server to your customer's browser, insert the following JavaScript into your checkout page. Note that a version of the jQuery JavaScript library must be loaded by your checkout page before loading the PayPage client JavaScript library.

NOTE: The URL in this example script (in red) should only be used in the certification and testing environment. Before using your checkout page with PayPage in a production environment, replace the certification URL with the production URL (contact your Implementation Consultant for the appropriate production URL).

```
<head>
...
<script
   src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
type="text/javascript">
  </script>
<script
   src="https://request-prelive.np-securepaypage-litle.com/LitlePayPage/litle-api2.js"
type="text/javascript">
  </script>
...
</head>
```

Do not use this URL in a production environment. Contact Implementation for the appropriate production URL.

NOTE: This example uses a Google-hosted version of the jQuery JavaScript library. You may choose to host the library locally. We recommend using version 1.4.2 or higher.

2.1.3 Specifying the PayPage API Request Fields

To specify the PayPage API request fields, add four hidden request fields to your checkout form for `paypageId` (a unique number assigned by Implementation), `merchantTxnId`, `orderId`, and `reportGroup` (LitleXML elements). You have flexibility in the naming of these fields.

NOTE: The values for either the `merchantTxnId` or the `orderId` must be unique so that we can use these identifiers for reconciliation or troubleshooting.

The values for `paypageId` and `reportGroup` will likely be constant in the HTML. The value for the `orderId` passed to the PayPage API can be generated dynamically.

```
<form
  ...
  <input type="hidden" id="request$paypageId" name="request$paypageId"
value="a2y4o6m8k0"/>
  <input type="hidden" id="request$merchantTxnId" name="request$merchantTxnId"
value="987012"/>
  <input type="hidden" id="request$orderId" name="request$orderId" value="order_123"/>
  <input type="hidden" id="request$reportGroup" name="request$reportGroup"
value="*merchant1500"/>
  ...
</form>
```

2.1.4 Specifying the PayPage API Response Fields

To specify the PayPage API Response fields, add seven hidden response fields on your checkout form for storing information returned by PayPage: `paypageRegistrationId`, `bin`, `code`, `message`, `responseTime`, `type`, and `litleTxnId`. You have the flexibility in the naming of these fields.

```
<form
  ...
  <input type="hidden" id="response$paypageRegistrationId"
name="response$paypageRegistrationId" readOnly="true" value=""/>
  <input type="hidden" id="response$bin" name="response$bin"  readOnly="true"/>
  <input type="hidden" id="response$code" name="response$code"  readOnly="true"/>
  <input type="hidden" id="response$message" name="response$message"
readOnly="true"/>
  <input type="hidden" id="response$responseTime" name="response$responseTime"
readOnly="true"/>
  <input type="hidden" id="response$type" name="response$type"  readOnly="true"/>
  <input type="hidden" id="response$litleTxnId" name="response$litleTxnId"
readOnly="true"/>
  <input type="hidden" id="response$firstSix" name="response$firstSix"
readOnly="true"/>
  <input type="hidden" id="response$lastFour" name="response$lastFour"
readOnly="true"/>
  ...
</form>
```

2.1.5 Handling the Mouse Click

In order to call the PayPage JavaScript API on the checkout form when your customer clicks the submit button, you must add a jQuery selector to handle the submission `click` JavaScript event. The addition of the `click` event creates a PayPage Request and calls `sendToLitle`.

The `sendToLitle` call includes a timeout value in milliseconds. We recommend a timeout value of 5000 (5 seconds).

NOTE: The URL in this example script (in red) should only be used in the certification and testing environment. Before using your checkout page with PayPage in a production environment, replace the certification URL with the production URL (contact your Implementation Consultant for the appropriate production URL).

```
<head>
...
  <script>
  ...
$("#submitId").click(
      function(){
      setLitleResponseFields({"response":"", "message":""});

      var litleRequest = {
       "paypageId" : document.getElementById("request$paypageId").value,
       "reportGroup" : document.getElementById("request$reportGroup").value,
       "orderId" : document.getElementById("request$orderId").value,
       "id" : document.getElementById("request$merchantTxnId").value,
       "url" : "https://request-prelive.np-securepaypage-litle.com"

      };
      new LitlePayPage().sendToLitle(litleRequest, formFields, submitAfterLitle,
onErrorAfterLitle, timeoutOnLitle, 5000);
            return false;

  ...
  </script>
...
</head>
```

Do not use this URL in a production environment. Contact Implementation for the appropriate production URL.

2.1.6 Intercepting the Checkout Form Submission

Without the PayPage implementation, order data is sent to your system when the submit button is clicked. With the PayPage feature, a request must be sent to our server to retrieve the PayPage Registration ID for the card number before the order is submitted to your system. To intercept the checkout form, you change the input type from `submit` to `button`. (The checkout button is built inside of a `<script>/<noscript>` pair, but the `<noscript>` element uses a message to alert the customer instead of providing a default `submit`.)

```
<BODY>
...
  <table>
  ...
  <tr><td></td><td align="right">
    <script>
      document.write('<button type="button" id="submitId" onclick="callLitle()">Check
out with paypage</button>');
    </script>
    <noscript>
      <button type="button" id="submitId">Enable JavaScript or call us at
555-555-1212</button></noscript>
    </td></tr>

...
</table>
...
</BODY>
```

2.1.7 Handling Callbacks for Success, Failure, and Timeout

Your checkout page must include instructions on what methods we should use to handle callbacks for success, failure, and timeout events. Add the code in the following three sections to achieve this.

2.1.7.1 Success Callbacks

The **success** callback stores the responses in the hidden form response fields and submits the form. The card number is scrubbed from the submitted form, and all of the hidden fields are submitted along with the other checkout information.

```
<head>
  ...
  <script>
  ...
function setLitleResponseFields(response) {
  document.getElementById('response$code').value = response.response;
  document.getElementById('response$message').value = response.message;
  document.getElementById('response$responseTime').value = response.responseTime;
  document.getElementById('response$litleTxnId').value = response.litleTxnId;
  document.getElementById('response$type').value = response.type;
  document.getElementById('response$firstSix').value = response.firstSix;
  document.getElementById('response$lastFour').value = response.lastFour;
```

```
}
function submitAfterLitle (response) {
  setLitleResponseFields(response);
  document.forms['fCheckout'].submit();
  }
  ...
  </script>
  ...
</head>
```

2.1.7.2 Failure Callbacks

There are two types of failures that can occur when your customer enters an order: validation (user) errors, and system (non-user) errors (see Table 1-2, "PayPage-Specific Response Codes Received in Browser or Mobile Device" on page 11). The **failure** callback stops the transaction for non-user errors and nothing is posted to your order handling system.

NOTE: When there is a timeout or you receive a validation-related error response code, be sure to submit enough information to your order processing system to identify transactions that could not be completed. This will help you monitor problems with the PayPage Integration and also have enough information for debugging.

You have flexibility in the wording of the error text.

```
<head>
...
   <script>
   ...
   function onErrorAfterLitle (response) {
     setLitleResponseFields(response);
     if(response.response == '871') {
       alert("Invalid card number.  Check and retry. (Not Mod10)");
     }
     else if(response.response == '872') {
       alert("Invalid card number.  Check and retry. (Too short)");
     }
     else if(response.response == '873') {
       alert("Invalid card number.  Check and retry. (Too long)");
     }
     else if(response.response == '874') {
       alert("Invalid card number.  Check and retry. (Not a number)");
     }
     else if(response.response == '875') {
       alert("We are experiencing technical difficulties. Please try again later or call
555-555-1212");
     }
     else if(response.response == '876') {
       alert("Invalid card number.  Check and retry. (Failure from Server)");
     }
     else if(response.response == '881') {
       alert("Invalid card validation code.  Check and retry. (Not a number)");
     }
```

```
      else if(response.response == '882') {
        alert("Invalid card validation code.  Check and retry. (Too short)");
      }
      else if(response.response == '883') {
        alert("Invalid card validation code.  Check and retry. (Too long)");
      }
      else if(response.response == '889') {
        alert("We are experiencing technical difficulties. Please try again later or call
 555-555-1212");
     }
 return false;
}
 ...
 </script>
...
</head>
```

2.1.7.3 Timeout Callbacks

The **timeout** callback stops the transaction and nothing is posted to your order handling system.

Timeout values are expressed in milliseconds and defined in the `sendToLitle` call, described in the section, Handling the Mouse Click on page 17. We recommend a timeout value of 5000 (5 seconds).

You have flexibility in the wording of the timeout error text.

```
<head>
...
  <script>
  ...
  function timeoutOnLitle () {
    alert("We are experiencing technical difficulties.  Please try again later or
call 555-555-1212 (timeout)");
  }
  ...
</script>
...
</head>
```

2.1.8 Detecting the Availability of the PayPage API

In the event that the `litle-api2.js` cannot be loaded, add the following to detect availability. You have flexibility in the wording of the error text.

```
</BODY>
...
<script>
  function callLitle() {
    if(typeof LitlePayPage !== 'function') {
      alert("We are experiencing technical difficulties.  Please try again later or
```

```
call 555-555-1212 (API unavailable)" );
</script>
...
</HTML>
```

A full HTML code example of a simple checkout page integrated with PayPage is shown in Appendix A, "Code Samples and Other Information".

2.2 Integrating Vantiv-Hosted iFrame into your Checkout Page

This section provides information and instructions for integrating the Vantiv-Hosted iFrame PayPage solution into your checkout page.

2.2.1 Integration Steps

Integrating the iFrame into you checkout page includes the following steps, described in the sections to follow. For a full HTML code example a Vantiv-Hosted iFrame PayPage implementation, see the HTML Example for Hosted iFrame-Integrated Checkout Page on page 64.

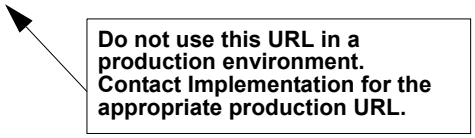
1. Loading the PayFrame
2. Configuring the iFrame
3. Calling the iFrame for the PayPage Registration ID
4. Handling Callbacks

NOTE: **The URL in this example (in red) should only be used in the certification and testing environment. Before using your checkout page with PayPage in a production environment, replace the certification URL with the production URL (contact your Implementation Consultant for the appropriate production URL).**

2.2.2 Loading the PayFrame

To load the PayFrame from the PayPage application server to your customer's browser, insert the following script tag into your checkout page:

```
<script src="https://request-prelive.np-securepaypage-litle.com/LitlePayPage/js
/payframe-client.min.js"></script>
```



2.2.3 Configuring the iFrame

To configure the iFrame after the page is loaded, you specify the required properties listed in Table 2-1 (other properties shown in the example below, are optional). You define a callback for errors, time-outs, and to retrieve the `paypageRegistrationId`. In this example, this is called `payframeClientCallback`.

```
$( document ).ready(function() {
  var configure = {
    "paypageId":document.getElementById("request$paypageId").value,
    "style":"test",
    "reportGroup":document.getElementById("request$reportGroup").value,
    "timeout":document.getElementById("request$timeout").value,
    "div": "payframe",
    "callback": payframeClientCallback,
    "showCvv": true,
    "months": {
      "1":"January",
      "2":"February",
      "3":"March",
      "4":"April",
      "5":"May",
      "6":"June",
      "7":"July",
      "8":"August",
      "9":"September",
      "10":"October",
      "11":"November",
      "12":"December"
    },
    "numYears": 8,
    "tooltipText": "A CVV is the 3 digit code on the back of your Visa, MasterCard and Discover or a
4 digit code on the front of your American Express",
    "tabIndex": {
      "cvv":1,
      "accountNumber":2,
      "expMonth":3,
      "expYear":4
    },
    "placeholderText": {
      "cvv":"CVV",
      "accountNumber":"Account Number"
    }
  };
  if(typeof LitlePayframeClient === 'undefined') {
      alert("We are experiencing technical difficulties.  Please try again or call us to complete
your order");
  //You may also want to submit information you have about the consumer to your servers to
facilitate debugging like customer ip address, user agent, and time
  }
  else {
  var payframeClient = new LitlePayframeClient(configure);
  payframeClient.autoAdjustHeight();
});
```

TABLE 2-1 Common Properties

Property	Description
paypageId	(*Required*) The unique number assigned by Implementation.
style	(*Required*) The CSS filename (excluding the '.css'). For example, if the style sheet filename is `mysheet1.css`, the value for this property is `mysheet1`.
reportGroup	(*Required*) The LitleXML required attribute that defines under which merchant sub-group this transaction will be displayed in iQ Reporting and Analytics.
timeout	(*Required*) The number of milliseconds before a transaction times out and the timeout callback in invoked. Vantiv recommends a value of 5000 (5 seconds).
div	(*Required*) The ID of the HTML `div` element where our iFrame is embedded as innerHTML.

callback	(*Required*) The function element that our iFrame calls with a single parameter representing a JSON dictionary. The keys in the callback are:
	`*paypageRegistrationId` `*orderId`
	`*bin` `*response`
	`*type` `*responseTime`
	`*firstSix` `*message`
	`*lastFour` `*reportGroup`
	`*expDate` `*id`
	`*litleTxnId` `*timeout`

height	(*Optional*) The height (in pixels) of the iFrame. There are three options: • You can pass `height` as an optional parameter when configuring the client. • You can call `autoAdjustHeight` in the client to tell the iFrame to adjust the height to exactly the number of pixels needed to display everything in the iFrame without displaying a vertical scroll bar (recommended). • You can ignore `height`. The iFrame may display a vertical scroll bar, depending upon your styling of the `div` containing the iFrame.

2.2.4 Calling the iFrame for the PayPage Registration ID

After your customer clicks the Submit/Complete Order button, your checkout page must call the iFrame to get a PayPage Registration ID. In the `onsubmit` event handler of your button, add code to call PayPage to get a Registration ID for the credit card and CVV. Include the parameters listed in Table 2-2.

```
document.getElementById("fCheckout").onsubmit = function(){
  var message = {
    "id":document.getElementById("request$merchantTxnId").value,
    "orderId":document.getElementById("request$orderId").value
  };
  payframeClient.getPaypageRegistrationId(message);
  return false;
};
```

TABLE 2-2 Event Handler Parameters

Parameter	Description
`id`	The LitleXML required attribute that assigned by the presenter and mirrored back in the response.
`orderId`	The merchant-assigned unique value representing the order in your system.

2.2.5 Handling Callbacks

After the iFrame has received the `paypageRegistrationId`, or has received an error or timed out, the iFrame calls the callback specified when the client was constructed. In your callback, you can determine success or failure by inspecting `response.response` (870 indicates success). You can check for a timeout by inspecting `response.timeout` (if it is defined, a timeout has occurred).

NOTE: When there is a timeout or you receive a validation-related error response code, be sure to submit enough information (for example, customer IP address, user agent, and time) to your order processing system to identify transactions that could not be completed. This will help you monitor problems with the PayPage Integration and also have enough information for debugging.

```
var payframeClientCallback = function(response) {
  if (response.timeout) {
  alert("We are experiencing technical difficulties.  Please try again or call us to complete your
order");
    //You may also want to submit information you have about the consumer to your servers to
facilitate debugging like customer ip address, user agent, and time
  }
  else {
    document.getElementById('response$code').value = response.response;
    document.getElementById('response$message').value = response.message;
    document.getElementById('response$responseTime').value = response.responseTime;
    document.getElementById('response$reportGroup').value = response.reportGroup;
    document.getElementById('response$merchantTxnId').value = response.id;
    document.getElementById('response$orderId').value = response.orderId;
    document.getElementById('response$litleTxnId').value = response.litleTxnId;
    document.getElementById('response$type').value = response.type;
    document.getElementById('response$lastFour').value = response.lastFour;
    document.getElementById('response$firstSix').value = response.firstSix;
    document.getElementById('paypageRegistrationId').value = response.paypageRegistrationId;
    document.getElementById('bin').value = response.bin;
    document.getElementById('response$expMonth').value = response.expMonth;
    document.getElementById('response$expYear').value = response.expYear;
    if(response.response === "870") {
            //Submit the form
    }
    else {
            alert("We are experiencing technical difficulties.  Please try again or call us to
complete your order");
        //You may also want to submit the litleTxnId and response received, plus information you
have about the consumer to your servers to facilitate debugging like customer ip address, user agent
and time
          }
        }
      };
```

2.2.5.1 Handling Errors

In case of errors in the iFrame, the iFrame adds an error class to the field that had the error. You can use those classes in the CSS you give Vantiv Implementation to provide error styles. The codes correspond to the response codes outlined in PayPage-Specific Response Codes on page 11.

- In case of error on the **accountNumber** field, these classes are added to the `div` in the iFrame with the existing class `numberDiv`.
  - `error-871`
  - `error-872`
  - `error-874`
  - `error-876`
- In case of error on the **cvv** field, these classes are added to the `div` in the iFrame with the existing class `cvvDiv`.
  - `error-881`
  - `error-882`

In either case, the callback is still invoked. When the input field with the error receives the focus event, we clear the error classes. Some sample CSS to indicate an error given these classes is as follows:

```
.error-871::before {
  content: "Account number not Mod10";
}
.error-871>input {
  background-color:red;
}
```

2.3 Integrating Mobile API Into Your Mobile Application

This section provides instructions for integrating the PayPage feature into your native mobile application. Unlike the PayPage browser checkout page solution, the native mobile application does not interact with the PayPage JavaScript in a browser. Instead, you use an HTTP POST in a native mobile application to send account numbers to Vantiv and receive a PayPage Registration ID in the response.

2.3.1 Creating the POST Request

You structure your POST request as shown in the Sample Request. Use the components listed in Table 2-3.

TABLE 2-3 POST Headers, Parameters, and URL

Component	Element	Description
Headers (optional)	`Content-Type: application/x-www-form-urlencoded`	
	`Host: request-prelive.np-securepaypage-litle.com/LitlePayPage/paypage`	
	`User-Agent: Litle/1.0 CFNetwork/459 Darwin/10.0.0.d3`	
Parameters (required)	**`paypageId`**	The unique number assigned by Implementation.
	`reportGroup`	The LitleXML required attribute that defines under which merchant sub-group this transaction will be displayed in iQ Reporting and Analytics.
	`orderId`	The merchant-assigned unique value representing the order in your system.
	`id`	The LitleXML required attribute that assigned by the presenter and mirrored back in the response.
	`accountNumber`	The 13-25-digit card account number. (Not used in Apple Pay transactions.)
(optional)	**`cvv`**	The card validation number, either the CVV2 (Visa), CVC2 (MasterCard), or CID (American Express and Discover) value. (Not used in Apple Pay transactions.)
URL	`https://request-prelive.np-securepaypage-litle.com/LitlePayPage/paypage`	

NOTE: The URL in this example script (in red) should only be used in the certification and testing environment. Before using your checkout page with PayPage in a production environment, replace the certification URL with the production URL (contact your Implementation Consultant for the appropriate production URL).

2.3.1.1 Sample Request

The following is an example POST to request a PayPage registration ID:

```
$ curl --verbose -H "Content-Type: application/x-www-form-urlencoded" -H "Host:
request-prelive.np-securepaypage-litle.com/LitlePayPage/paypage" -H
"User-Agent: Litle/1.0 CFNetwork/459 Darwin/10.0.0.d3" -d"paypageId=a2y4o6m8k0&
reportGroup=*merchant1500&orderId=PValid&id=12345&accountNumber=ACCOUNT_NUMBER&cvv=123"
https://request-prelive.np-securepaypage-litle.com/LitlePayPage/paypage
```

2.3.1.2 Sample Response

The response received in the body of the POST response is a JSON string similar to the following:

```
{"bin":"410000","firstSix":"410000","lastFour":"0001","paypageRegistrationId":"amNDNkpWck
VGNFJoRmdNeXJUOHl4Skh1TTQ1Z0t6WE9TYmdqdjBJT0F5N28zbUpxdlhGazZFdmlCSzdTN3ptKw\u003d\u003d"
,"type":"VI","id":"12345","litleTxnId":"83088059521107596","message":"Success","orderId":
"PValid","reportGroup":"*merchant1500","response":"870","responseTime":"2014-02-07T17:04:
04"}
```

Table 2-4 lists the parameters included in the response.

TABLE 2-4 Parameters Returned in POST Response

Parameter	Description
`paypageRegistrationId`	The temporary identifier used to facilitate the mapping of a token to a card number.
`reportGroup`	(Mirrored back from the request) The LitleXML required attribute that defines under which merchant sub-group this transaction will be displayed in iQ Reporting and Analytics.
`orderId`	(Mirrored back from the request) The merchant-assigned unique value representing the order in your system.

TABLE 2-4 Parameters Returned in POST Response (Continued)

Parameter	Description
`id`	(Mirrored back from the request) The LitleXML required attribute that assigned by the presenter and mirrored back in the response.
`litleTxnId`	The automatically-assigned unique transaction identifier.
`firstSix`	(Mirrored back from the request) The first six digits of the credit card number.
`lastFour`	(Mirrored back from the request) The last four digits of the credit card number.

2.3.2 PayPage Support for Apple Pay™

NOTE: This section is an excerpt from the Vantiv eCommerceTechnical Publication, *Vantiv eCommerce Solution for Apple Pay*. Refer to the full document for further information.

Vantiv supports Apple Pay for in-app purchases initiated on the iPhone 6. Merchants wishing to allow Apple Pay transactions from their native iOS mobile applications will have to build the capability to make secure purchases using Apple Pay into their mobile application. The operation of Apple Pay on the iPhone 6 is relatively simple, but requires either the development of new native iOS applications or the modification of your existing applications that include the use of the Apple PassKit Framework, and the handling of the encrypted data returned to your application by Apple Pay.

NOTE: If you subscribe to both Vault tokenization and Apple Pay, Vantiv will tokenize Apple Pay token values to ensure a consistent token value is returned. As a result, tokenized value returned in the response is based off the Apple Pay token, not the original PAN value. Format preserving components of the Vault token value such as the Last-four and BIN will be from the Apple Pay token, not the PAN.

2.3.2.1 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 LitleXML transaction using that ID.

The steps that occur when a consumer initiates an Apple Pay purchase using your mobile application are detailed below and shown in Figure 2-1.

1. When the consumer selects the Apple Pay option from your application, your application makes use of the Apple PassKit Framework 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, 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) to your application.
4. Your native iOS application sends the PKPaymentToken to our secure server via an HTTPS POST (see Creating a POST Request for an Apple Pay Transaction on page 33) and PayPage returns a PayPage Registration ID.
5. Your native iOS application forwards the transaction data along with the PayPage Registration ID to your order processing server, as it would with any PayPage transaction.
6. Your server constructs/submits a standard LitleXML Authorization/Sale transaction using the PayPage Registration ID.
7. Using the private key, Vantiv decrypts the PKPaymentToken associated with the PayPage 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.

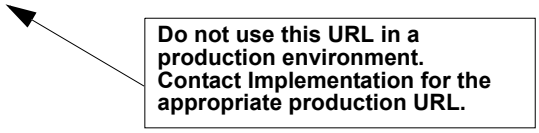
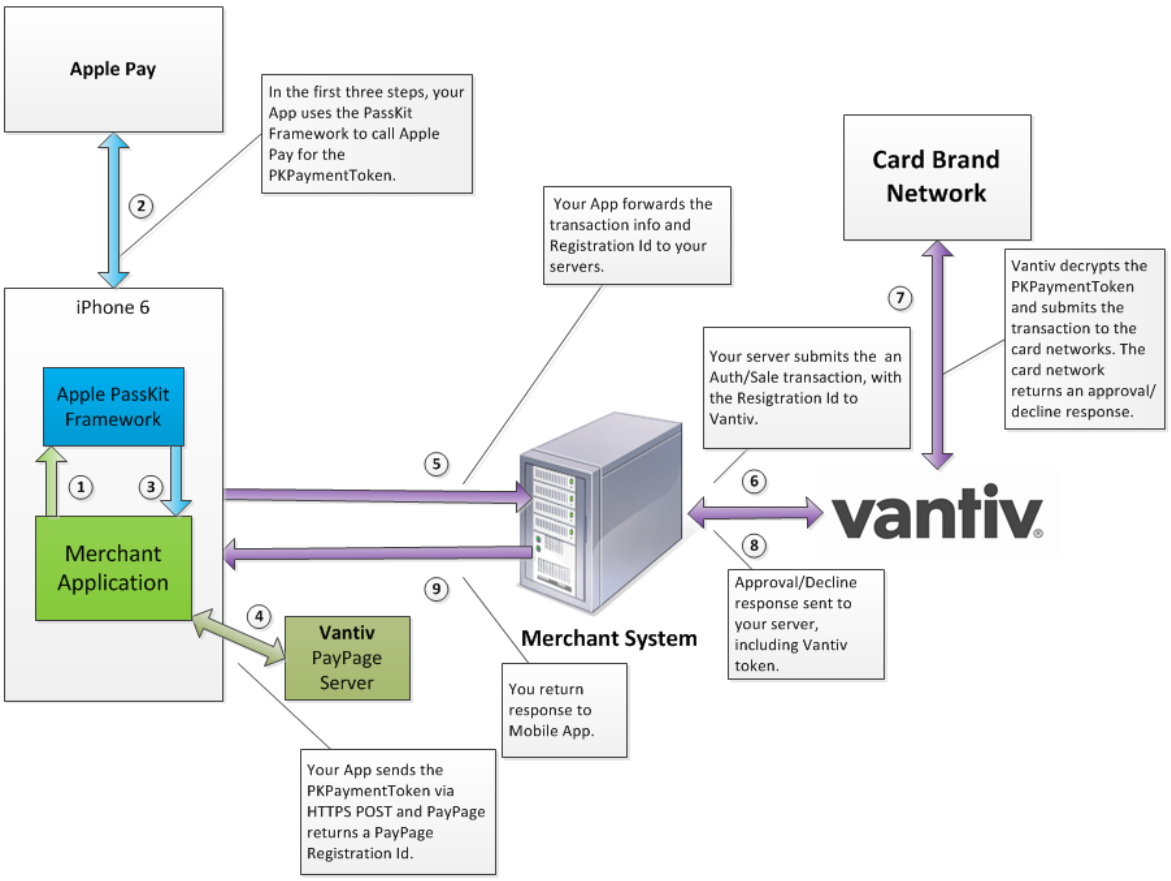


FIGURE 2-1 Data/Transaction Flow using the Vantiv Mobile API for Apple Pay



2.3.2.2 Creating a POST Request for an Apple Pay Transaction

Construct your HTTPS POST as detailed in Creating the POST Request on page 28, using the components listed in the Table 2-3 as well as those listed in Table 2-5 (all required). See the Sample Apple Pay POST Request and Sample Apple Pay POST Response below.

TABLE 2-5 Vantiv Mobile API for Apple Pay HTTPS POST Required Components

Parameter Name	Description
`applepay.data`	Payment data dictionary, Base64 encoded as a string. Encrypted Payment data.
`applepay.signature`	Detached PKCS #7 signature, Base64 encoded as string. Signature of the payment and header data.
`applepay.version`	Version information about the payment token.
`applepay.header.applicationData`	SHA-256 hash, Base64 encoded as a string. Hash of the applicationData property of the original PKPaymentRequest.
`applepay.header.ephemeralPublicKey`	X.509 encoded key bytes, Base64 encoded as a string. Ephemeral public key bytes.
`applepay.header.publicKeyHash`	SHA-256 hash, Base64 encoded as a string. Hash of the X.509 encoded public key bytes of the merchant's certificate.
`applepay.header.transactionId`	Hexademical identifier, as a string. Transaction identifier, generated on the device.

2.3.2.3 Sample Apple Pay POST Request

The following is an example POST to request a PayPage registration ID for Apple Pay:

```
curl --verbose -H "Content-Type: application/x-www-form-urlencoded" -H "Host:MerchantApp"
-H "User-Agent:Litle/1.0 CFNetwork/459 Darwin/10.0.0.d3"
-d"paypageId=a2y4o6m8k0&reportGroup=*merchant1500&orderId=PValid&id=1234&applepay.data=HT
897mACd%2F%2FTpWe10A5y9RmL5UfboTiDIvjni3zWFtyy8dtv72RJL1bk%2FU4dTDlrq1T1V2l0TSnI%0APLdOnn
HBO51bt9Ztj9odDTQ5LD%2F4hMZTQj3lBRvFOtTtjp9ysBAsydgjEjcCcbnkx7dCqgnwgzuz%0Ay7bX%2B5Fo8a8R
KqoprkDPwIMWOC9yWe7MQw%2FboM5NY2QtIcIvzbLFcYUxndYTg0IXNBHNzsvUOjmw%0AvEnMhXxeCH%2BC4KoC6M
EsAGK5rH1T5dSvTZzHF5c12dpsqdI73%2FBk6qEcdlT7gJKVmyDQC%2FNFxJ0X%0AF993Of6ejQDJq6BZsz8X7kYC
yJdI%2FPFJPZp4e3L%2FtCsBDUTJAgFLt2xF8HWaPoW8psILOGCCvJQm%0ATR1m7ODtSChaWOb7eYm1BpNiD3wkCH
8nmIMrlnt3KP4SeQ%3D%3D&applepay.signature=MIAGCSqGSIb3DQEHAqCAMIACAQExDzANBglghkgBZQMEAgE
FADCABgkqhkiG9w0BBwEAAKCAMIICvzCCAmWgAwIBAgIIQpCV6UIIb4owCgYIKoZIzj0EAwIwejEuMCwGA1UEAwwl
QXBwbGUgQXBwbGljYXRpb24gSW50ZWdyYXRpb24gQ0EgLSBHMzEmMCQGA1UECwwdQXBwbGUgQ2VydGlmaWNhdGlvb
iBBdXRob3JpdHkxEzARBgNVBAoMCkFwcGxlIEluYy4xCzAJBgNVBAYTAlVTMB4XDTE0MDUwODAxMjMzOVoXDTE5MD
UwNzAxMjMzOVowXzElMCMGA1UEAwwcZWNjLXNtcC1icm9rZXItc2lnbl9VQzQtUFJPRDEUMBIGA1UECwwLaU9TIFN
5c3RlbXMxEzARBgNVBAoMCkFwcGxlIEluYy4xCzAJBgNVBAYTAlVTMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE
whV37evWx7Ihj2jdcJChIY3HsL1vLCg9hGCV2Ur0pUEbg0IO2BHzQH6DMx8cVMP36zIg1rrV1O%2F0komJPnwPE6O
B7zCB7DBFBggrBgEFBQcBAQQ5MDcwNQYIKwYBBQUHMAGGKWh0dHA6Ly9vY3NwLmFwcGxlLmNvbS9vY3NwMDQtYXBw
bGVhaWNhMzAxMB0GA1UdDgQWBBSUV9tv1XSBhomJdi9%2BV4UH55tYJDAMBgNVHRMBAf8EAjAAMB8GA1UdIwQYMBa
AFCPyScRPk%2BTvJ%2BbE9ihsP6K7%2FS5LMDQGA1UdHwQtMCswKaAnoCWGI2h0dHA6Ly9jcmwuYXBwbGUuY29tL2
FwcGxlYWljYTMuY3JsMA4GA1UdDwEB%2FwQEAwIHgDAPBgkqhkiG92NkBh0EAgUAMAoGCCqGSM49BAMCA0gAMEUCI
QCFGdtAk%2B7wXrBV7jTwzCBLE%2BOcrVL15hjif0reLJiPGgIgXGHYYeXwrn02Zwcl5TT1W8rIqK0QuIvOnO1THC
bkhVowggLuMIICdaADAgECAghJbS%2B%2FOpjalzAKBggqhkjOPQQDAjBnMRswGQYDVQQDDBJBcHBsZSBSb290IEN
BIC0gRzMxJjAkBgNVBAsMHUFwcGxlIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MRMwEQYDVQQKDApBcHBsZSBJbmMu
MQswCQYDVQQGEwJVUzAeFw0xNDA1MDYyMzQ2MzBaFw0yOTA1MDYyMzQ2MzBaMHoxLjAsBgNVBAMMJUFwcGxlIEFwc
GxpY2F0aW9uIEludGVncmF0aW9uIENBIC0gRzMxJjAkBgNVBAsMHUFwcGxlIENlcnRpZmljYXRpb24gQXV0aG9yaX
R5MRMwEQYDVQQKDApBcHBsZSBJbmMuMQswCQYDVQQGEwJVUzBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABPAXEYQ
Z12SF1RpeJYEHduiAou%2Fee65N4I38S5PhM1bVZls1riLQl3YNIk57ugj9dhfOiMt2u2ZwvsjoKYT%2FVEWjgfcw
gfQwRgYIKwYBBQUHAQEEOjA4MDYGCCsGAQUFBzABhipodHRwOi8vb2NzcC5hcHBsZS5jb20vb2NzcDA0LWFwcGxlc
m9vdGNhZzMwHQYDVR0OBBYEFCPyScRPk%2BTvJ%2BbE9ihsP6K7%2FS5LMA8GA1UdEwEB%2FwQFMAMBAf8wHwYDVR
0jBBgwFoAUu7DeoVgziJqkipnevr3rr9rLJKswNwYDVR0fBDAwLjAsoCqgKIYmaHR0cDovL2NybC5hcHBsZS5jb20
vYXBwbGVyb290Y2FnMy5jcmwwDgYDVR0PAQH%2FBAQDAgEGMBAGCiqGSIb3Y2QGAg4EAgUAMAoGCCqGSM49BAMCA2
cAMGQCMDrPcoNRFpmxhvs1w1bKYr%2F0F%2B3ZD3VNoo6%2B8ZyBXkK3ifiY95tZn5jVQQ2PnenC%2FgIwMi3VRCG
wowV3bF3zODuQZ%2F0XfCwhbZZPxnJpghJvVPh6fRuZy5sJiSFhBpkPCZIdAAAxggFfMIIBWwIBATCBhjB6MS4wLA
YDVQQDDCVBcHBsZSBBcHBsaWNhdGlvbiBJbnRlZ3JhdGlvbiBDQSAtIEczMSYwJAYDVQQLDB1BcHBsZSBDZXJ0aWZ
pY2F0aW9uIEF1dGhvcml0eTETMBEGA1UECgwKQXBwbGUgSW5jLjELMAkGA1UEBhMCVVMCCEKQlelCCG%2BKMA0GCW
CGSAFlAwQCAQUAoGkwGAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEHATAcBgkqhkiG9w0BCQUxDxcNMTQxMDAzMjE1NjQ
zWjAvBgkqhkiG9w0BCQQxIgQgg8i4X6yRAU7AXS1lamCf02UIQlpUvNPToXUaamsFUT8wCgYIKoZIzj0EAwIERzBF
AiBe17NGTuuk%2BW901k3Oac4Z90PoMhN1qRqnij9KNEb%2FXAIhALELZyDWw0fQM8t0pXO86gg9xXFz424rEMlJ0
1TM1VxhAAAAAAAA&applepay.version=EC_v1&applepay.header.applicationData=496461ea64b50527d2
d792df7c38f301300085dd463e347453ae72debf6f4d14&applepay.header.ephemeralPublicKey=MFkwEwY
HKoZIzj0CAQYIKoZIzj0DAQcDQgAEarp8xOhLX9QliUPS9c54i3cqEfrJD37NG75ieNxncOeFLkjCk%2FBn3jVxHl
ecRwYqe%2BAWQxZBtDyewaZcmWz5lg%3D%3D&applepay.header.publicKeyHash=zoV5b2%2BmqnMIxU9avTeq
Wxc7OW3fnKXFxyhY0cyRixU%3D&applepay.header.transactionId=23e26bd8741fea9e7a4d78a69f4255b3
15d39ec14233d6f1b32223d1999fb99f" https://request-prelive.np-securepaypage-litle.
com/LitlePayPage/paypage
```

2.3.2.4 Sample Apple Pay POST Response

The response received in the body of the POST response is a JSON string similar to the following:

{"bin":"410000","firstSix":"410000","lastFour":"0001","paypageRegistrationId":"S0ZBUURMTlZkMTgrbW1IL3BZVFFmaDh0M0hjdDZ5RXcxQzRQUkJRKzdVc3JURXp0N0JBdmhDN05aTllUQU5rY1RCMDhLNXg2clI0cDV3Sk5vQmlPTjY3V2plbDVac0lqd0FkblYwVTdQWms9","type":"VI","id":"1234","litleTxnId":"8282662615343150 9","message":"Success","orderId":"PValid","reportGroup":"*merchant1500","response":"870","responseTime":"2015-01-19T18:35:27"}

2.4 Collecting Diagnostic Information

In order to assist Vantiv in determining the cause of failed PayPage transactions (and avoid potential lost sales), please collect the following diagnostic information when you encounter a failure, and provide it to your **Implementation Consultant** if you are currently in the testing and certification process, or your **Customer Experience Manager** if you are currently in production.

- Error code returned and reason for the failure:
  - JavaScript was disabled on the customer's browser.
  - JavaScript could not be loaded.
  - JavaScript was loaded properly, but the `sendToLitle` call did not return a response, or timed out (JavaScript API and Mobile API only).
  - JavaScript was loaded properly, but the `sendToLitle` call returned a response code indicating an error (JavaScript API and Mobile API only).
  - JavaScript was loaded properly, but the call to construct the `PayFrameClient` failed (iFrame only).
  - JavaScript was loaded properly, but the `getPaypageRegistrationId` call failed (iFrame only).
- The `orderId` and `merchantTxnId` for the transaction.
- Where in the process the failure occurred.
- Information about the customer's browser, including the version.

For further information on methods for collecting diagnostic information, contact your Implementation Consultant if your are currently in the testing and certification process, or your Customer Experience Manager if you are currently in production.

2.5 LitleXML Transaction Examples When Using PayPage

This section describes how to format LitleXML transactions when using the PayPage feature of the Vault solution. These standard LitleXML transactions are submitted by your payment processing system after your customer clicks the submit button on your checkout page. Your payment processing system sends the transactions to Vantiv with the `<paypageRegistrationId>` from the response message, and the Vault maps the Registration ID to the token and card number, processing the payment as usual.

NOTE: **The PayPage Registration ID is a temporary identifier used to facilitate the mapping of a token to a card number, and consequently expires within 24 hours of issuance. If you do not submit an Authorization, Sale, or Register Token transaction containing the `<paypageRegistrationId>` within 24 hours, the system returns a response code of 878 - *Expired PayPage Registration ID,* and no token is issued.**

See LitleXML Elements for PayPage on page 83 for definitions of the PayPage-related elements used in these examples.

This section is meant as a supplement to the *Vantiv LitleXML Reference Guide*. Refer to the *Vantiv LitleXML Reference Guide* for comprehensive information on all elements used in these examples.

2.5.1 Transaction Types and Examples

This section contains examples of the following transaction types:

- Authorization Transactions
- Sale Transactions
- Register Token Transactions
- Force Capture Transactions
- Capture Given Auth Transactions
- Credit Transactions

For each type of transaction, only online examples are shown, however batch transactions for all the above transaction types are also supported when using the PayPage feature. See the *Vantiv LitleXML Reference Guide* for information on forming batch transactions.

2.5.2 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.

This section describes the format you must use for an Authorization request when using the PayPage feature, as well as the Authorization Response format.

2.5.2.1 Authorization Request Structure

You must structure an Authorization request as shown in the following examples when using PayPage.

```
<authorization id="Authorization Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Authorization Amount</amount>
  <orderSource>ecommerce</orderSource>
  <billToAddress>
  <shipFromPostalCode>
  <paypage>
    <paypageRegistrationId>Registation ID returned</paypageRegistrationId>
    <expDate>Card Expiration Date</expDate>
    <cardValidationNum>Card Validation Number</cardValidationNum>
  </paypage>
</authorization>
```

Example: Online Authorization Request

```
<litleOnlineRequest version="9.4" xmlns="http://www.litle.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>ecommerce</orderSource>
    <billToAddress>
      <name>John Smith</name>
```

```
      <addressLine1>100 Main St</addressLine1>
      <city>Boston</city>
      <state>MA</state>
      <zip>12345</zip>
      <email>jsmith@someaddress.com</email>
      <phone>555-123-4567</phone>
    </billToAddress>
    <paypage>
      <paypageRegistrationId>cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NNlYE4ZW5UTko4NU
  9KK3p1L1p1VzE4ZWVPQVlSUHNITG1JN2I0NzlyTg=</paypageRegistrationId>
      <expDate>1012</expDate>
      <cardValidationNum>000</cardValidationNum>
    </paypage>
  </authorization>
</litleOnlineRequest>
```

2.5.2.2 Authorization Response Structure

An Authorization response has the following structure:

```
<authorizationResponse id="Authorization Id" reportGroup="UI Report
Group" customerId="Customer Id">
  <litleTxnId>Transaction Id</litleTxnId>
  <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>
  <accountInformation>
  <fraudResult>
  <tokenResponse>
</authorizationResponse>
```

Example: Online Authorization Response

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

```
<litleOnlineResponse version="9.4" xmlns="http://www.litle.com/schema"
  response="0" message="Valid Format">
  <authorizationResponse id="834262" reportGroup="ABC Division"
  customerId="038945">
    <litleTxnId>969506</litleTxnId>
    <orderId>65347567</orderId>
    <response>000</response>
    <responseTime>2009-07-25T15:13:43</responseTime>
    <postDate>2009-07-25</postDate>
    <message>Approved</message>
    <authCode>123457</authCode>
    <fraudResult>
      <avsResult>11</avsResult>
      <cardValidationResult>P</cardValidationResult>
    </fraudResult>
    <tokenResponse>
      <litleToken>1111000100090005</litleToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </authorizationResponse>
</litleOnlineResponse>
```

2.5.3 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.

This section describes the format you must use for a sale request, as well as the format of the Sale Response.

2.5.3.1 Sale Request Structure

You must structure a Sale request as shown in the following examples when using PayPage:

```
<sale id="Authorization Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Authorization Amount</amount>
  <orderSource>ecommerce</orderSource>
  <billToAddress>
  <shipFromPostalCode>
  <paypage>
    <paypageRegistrationId>Registation ID returned</paypageRegistrationId>
    <expDate>Card Expiration Date</expDate>
    <cardValidationNum>Card Validation Number</cardValidationNum>
  </paypage>
</sale>
```

Example: Online Sale Request

```
<litleOnlineRequest version="9.4" xmlns="http://www.litle.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <sale id="834262" reportGroup="ABC Division" customerId="038945">
    <orderId>65347567</orderId>
    <amount>40000</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
```

```
      <name>John Smith</name>
      <addressLine1>100 Main St</addressLine1>
      <city>Boston</city>
      <state>MA</state>
      <zip>12345</zip>
      <email>jsmith@someaddress.com</email>
      <phone>555-123-4567</phone>
    </billToAddress>
    <paypage>
      <paypageRegistrationId>cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NNlYE4ZW5UTko4NU
  9KK3p1L1p1VzE4ZWVPQVlSUHNITG1JN2I0NzlyTg=</paypageRegistrationId>
      <expDate>1012</expDate>
      <cardValidationNum>000</cardValidationNum>
    </paypage>
  </sale>
</litleOnlineRequest>
```

2.5.3.2 Sale Response Structure

A Sale response has the following structure:

```
<SaleResponse id="Authorization Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <litleTxnId>Transaction Id</litleTxnId>
  <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>
  <accountInformation>
  <fraudResult>
  <tokenResponse>
</SaleResponse>
```

Example: Online Sale Response

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

```
<litleOnlineResponse version="9.4" xmlns="http://www.litle.com/schema"
  response="0" message="Valid Format">
  <saleResponse id="834262" reportGroup="ABC Division" customerId="038945">
    <litleTxnId>969506</litleTxnId>
    <orderId>65347567</orderId>
    <response>000</response>
    <responseTime>2009-07-25T15:13:43</responseTime>
    <postDate>2009-07-25</postDate>
    <message>Approved</message>
    <authCode>123457</authCode>
    <fraudResult>
      <avsResult>11</avsResult>
      <cardValidationResult>P</cardValidationResult>
    </fraudResult>
    <tokenResponse>
      <litleToken>1111000100090005</litleToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </saleResponse>
</litleOnlineResponse>
```

2.5.4 Register Token Transactions

The Register Token transaction enables you to submit a credit card number, eCheck account number, or in this case, a PayPage Registration Id to our system and receive a token in return.

2.5.4.1 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 a PayPage Registration Id.

When you submit the CVV2/CVC2/CID in a `registerTokenRequest`, our platform encrypts and stores the value on a temporary basis for later use in a tokenized Authorization or 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.

NOTE: **The use of the `<cardValidationNum>` element in the `<registertokenRequest>` only applies when you submit an `<accountNumber>` element.**

For PayPage Registration IDs:

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

For Credit Card and eCheck Register Token request structures, see the *Vantiv LitleXML Reference Guide*.

Example: Online Register Token Request - PayPage

```
<litleOnlineRequest version="9.4" xmlns="http://www.litle.com/schema"
  merchantId="100">
  <authentication>
    <user>userName</user>
    <password>password</password>
  </authentication>
    <registerTokenRequest id="99999" reportGroup="RG1">
```

```
    <orderId>F12345</orderId>
    <paypageRegistrationId>cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NNlYE4ZW5UTko4NU
9KK3p1L1p1VzE4ZWVPQVlSUHNITG1JN2I0NzlyTg=</paypageRegistrationId>
  </registerTokenRequest>
</litleoOnlineRequest>
```

2.5.4.2 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, eCheck account, or PayPage registration Id. The response for the will have one of the following structures.

Register Token response for PayPage Registration Ids (and Credit Cards):

```
<registerTokenResponse id="99999" reportGroup="RG1">
  <litleTxnId>Transaction ID</litleTxnId>
  <orderId>Order Id</orderId>
  <litleToken>Token</litleToken>
  <bin>BIN</bin>
  <type>Method of Payment</type>
  <response>Response Code</response>
  <responseTime>Response Time</responseTime>
  <message>Response Message</message>
</registerTokenResponse>
```

For eCheck Register Token response structures, see the *Vantiv LitleXML Reference Guide*.

Example: Online Register Token Response - PayPage

```
<litleOnlineResponse version="9.4" xmlns="http://www.litle.com/schema"
  id="123" response="0" message="Valid Format" litleSessionId="987654321">
    <registerTokenResponse id="99999" reportGroup="RG1">
      <litleTxnId>21122700</litleTxnId>
      <orderId>F12345</orderId>
      <litleToken>1111000100360002</litleToken>
      <bin>400510</bin>
      <type>VI</type>
      <response>801</response>
      <responseTime>2010-10-26T17:21:51</responseTime>
      <message>Account number was successfully registered</message>
    </registerTokenResponse>
</litleOnlineResponse>
```

2.5.5 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. You can use a `<paypageRegistrationID>` with a Force Capture transaction.

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.

2.5.5.1 Force Capture Request

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

```
<forceCapture id="Id" reportGroup="UI Report Group" customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Force Capture Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
  <paypage>
    <paypageRegistrationId>Registation ID returned</paypageRegistrationId>
    <expDate>Card Expiration Date</expDate>
    <cardValidationNum>Card Validation Number</cardValidationNum>
  </paypage>
</forceCapture>
```

Example: On-Line Force Capture Request

```
<litleOnlineRequest version="9.4" xmlns="http://www.litle.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <forceCapture  id="834262" reportGroup="ABC Division" customerId="038945">
    <orderId>65347567</orderId>
    <amount>40000</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
      <name>John Smith</name>
```

```
      <addressLine1>100 Main St</addressLine1>
      <city>Boston</city>
      <state>MA</state>
      <zip>12345</zip>
      <country>USA</country>
      <email>jsmith@someaddress.com</email>
      <phone>555-123-4567</phone>
    </billToAddress>
    <paypage>
      <paypageRegistrationId>cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NNlYE4ZW5UTko4NU
  9KK3p1L1p1VzE4ZWVPQVlSUHNITG1JN2I0NzlyTg=</paypageRegistrationId>
      <expDate>1012</expDate>
      <cardValidationNum>712</cardValidationNum>
    </paypage>
  </forceCapture>
</litleOnlineRequest>
```

2.5.5.2 Force Capture Response

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

```
<forceCaptureResponse id="Capture Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <litleTxnId>Transaction Id</litleTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date of Posting</postDate> (Online Only)
  <message>Response Message</message>
  <tokenResponse>
  <accountUpdater>
</forceCaptureResponse>
```

Example: Force Capture Response

```
<litleOnlineResponse version="9.4" xmlns="http://www.litle.com/schema"
  response="0" message="Valid Format">
  <forceCaptureResponse id="2" reportGroup="ABC Division"
   customerId="038945">
    <litleTxnId>1100030204</litleTxnId>
    <orderId>65347567</orderId>
```

```
    <response>000</response>
    <responseTime>2009-07-11T14:48:48</responseTime>
    <postDate>2009-07-11</postDate>
    <message>Approved</message>
    <tokenResponse>
      <litleToken>1111000100090005</litleToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </forceCaptureResponse>
</litleOnlineResponse>
```

2.5.6 Capture Given Auth Transactions

You can use a Capture Given Auth transaction with a <paypageRegistrationID> if the <litleTxnId> is unknown and the Authorization was processed using COMAAR data (**C**ard Number, **O**rder Id, **M**erchant Id, **A**mount, **A**pproval Code, and (Auth) **R**esponse Date).

2.5.6.1 Capture Given Auth Request

```
<captureGivenAuth id="Capture Given Auth Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <orderId>Order Id</orderId>
  <authInformation>
  <amount>Authorization Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
  <shipToAddress>
  <paypage>
    <paypageRegistrationId>Registation ID returned</paypageRegistrationId>
    <expDate>Card Expiration Date</expDate>
    <cardValidationNum>Card Validation Number</cardValidationNum>
  </paypage>
</captureGivenAuth>
```

Example: Online Capture Given Auth Request

```
<litleOnlineRequest version="9.4" xmlns="http://www.litle.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <captureGivenAuth id="834262" reportGroup="ABC Division"
   customerId="038945">
    <orderId>65347567</orderId>
    <authInformation>
      <authDate>2011-06-22</authDate>
      <authCode>111111</authCode>
    </authInformation>
    <amount>40000</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
      <name>John Smith</name>
      <addressLine1>100 Main St</addressLine1>
      <city>Boston</city>
      <state>MA</state>
      <zip>12345</zip>
      <country>USA</country>
      <email>jsmith@someaddress.com</email>
      <phone>555-123-4567</phone>
    </billToAddress>
    <paypage>
      <paypageRegistrationId>cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NNlYE4ZW5UTko4NU
  9KK3p1L1p1VzE4ZWVPQVlSUHNITG1JN2I0NzlyTg=</paypageRegistrationId>
      <expDate>1012</expDate>
      <cardValidationNum>000</cardValidationNum>
    </paypage>
  </captureGivenAuth>
</litleOnlineRequest>
```

2.5.6.2 Capture Given Auth Response

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

```
<captureGivenAuthResponse id="Capture Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <litleTxnId>Transaction Id</litleTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date of Posting</postDate> (Online Only)
  <message>Response Message</message>
  <tokenResponse>
</captureGivenAuthResponse>
```

Example: Online Capture Given Auth Response

```
<litleOnlineResponse version="9.4" xmlns="http://www.litle.com/schema"
  response="0" message="Valid Format">
  <captureGivenAuthResponse id="2" reportGroup="ABC Division"
   customerId="038945">
    <litleTxnId>1100030204</litleTxnId>
    <orderId>65347567</orderId>
    <response>000</response>
    <responseTime>2011-07-11T14:48:48</responseTime>
    <postDate>2011-07-11</postDate>
    <message>Approved</message>
    <tokenResponse>
      <litleToken>1111000100090005</litleToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </captureGivenAuthResponse>
</litleOnlineResponse>
```

2.5.7 Credit Transactions

The Credit transaction enables you to refund money to a customer. You can submit refunds against any of the following payment transactions using a <paypageRegistrationId>:

- Capture Given Auth Transactions
- Force Capture Transactions
- Sale Transactions

2.5.7.1 Credit Request Transaction

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

```
<credit id="Credit Id" reportGroup="UI Report Group" customerId="Customer Id">
  <orderId>Order Id</orderId>
  <amount>Authorization Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
  <paypage>
    <paypageRegistrationId>Registation ID returned</paypageRegistrationId>
    <expDate>Card Expiration Date</expDate>
    <cardValidationNum>Card Validation Number</cardValidationNum>
  </paypage>
  <customBilling>
  <enhancedData>
</credit>
```

Example: Online Credit Request Transaction

```
<litleOnlineRequest version="9.4" xmlns="http://www.litle.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <credit id="834262" reportGroup="ABC Division" customerId="038945">
    <orderId>65347567</orderId>
    <amount>40000</amount>
    <orderSource>ecommerce</orderSource>
    <billToAddress>
      <name>John Smith</name>
```

```
      <addressLine1>100 Main St</addressLine1>
      <city>Boston</city>
      <state>MA</state>
      <zip>12345</zip>
      <email>jsmith@someaddress.com</email>
      <phone>555-123-4567</phone>
    </billToAddress>
    <paypage>
      <paypageRegistrationId>cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NNlYE4ZW5UTko4NU
  9KK3p1L1p1VzE4ZWVPQVlSUHNITG1JN2I0NzlyTg=</paypageRegistrationId>
      <expDate>1012</expDate>
      <cardValidationNum>000</cardValidationNum>
    </paypage>
  </credit>
</litleOnlineRequest>
```

2.5.7.2 Credit Response

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

```
<creditResponse id="Credit Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <litleTxnId>Transaction Id</litleTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date of Posting</postDate> (Online Only)
  <message>Response Message</message>
  <tokenResponse>
</creditResponse>
```

Example: Online Credit Response

```
<litleOnlineResponse version="9.4" xmlns="http://www.litle.com/schema"
  response="0" message="Valid Format">
  <creditResponse customerId="038945" id="5" reportGroup="ABC Division">
    <litleTxnId>1100030204</litleTxnId>
    <orderId>452345234</orderId>
    <response>000</response>
    <responseTime>2009-08-11T14:48:48</responseTime>
    <postDate>2009-08-11</postDate>
```

```
      <message>Approved</message>
      <tokenResponse>
        <litleToken>1111000100090005</litleToken>
        <tokenResponseCode>801</tokenResponseCode>
        <tokenMessage>Account number was successfully registered</tokenMessage>
        <type>VI</type>
        <bin>402410</bin>
      </tokenResponse>
    </creditResponse>
  </litleOnlineResponse>
```

2.6 Testing and Certification

Vantiv requires successful certification testing for the PayPage transactions before you can use them in production. During certification testing, you will work through each required test scenario with an Implementation Consultant. This section provides the specific data you must use in your PayPage transactions when performing the required tests. Use of this data allows the validation of your transaction structure/syntax, as well as the return of a response file containing known data.

The testing process for PayPage includes browser and/or mobile application interaction, JavaScript interaction, as well as XML requests and responses. The PayPage Certification tests the following:

For browser-based checkout pages and **mobile applications**:

- A successful transaction
- LitleXML transaction requests and responses

For browser-based checkout pages only:

- The timeout period
- The error handler and JavaScript error codes

See the section, PayPage-Specific Response Codes on page 11 for definitions of the response codes.

2.6.1 Testing PayPage Transactions

To test PayPage transactions:

1. Verify that your checkout page or mobile application is coded correctly. See one of the following sections for more information:
   - Integrating Customer Browser JavaScript API Into Your Checkout Page on page 14 .
   - Integrating Vantiv-Hosted iFrame into your Checkout Page on page 22.
   - Integrating Mobile API Into Your Mobile Application on page 28.
2. Verify that you are using the appropriate URL (see Table 1-1, "PayPage Certification, Testing, and Production URLs" on page 9) for the testing and certification environment, for example:

   ```
   https://request-prelive.np-securepaypage-litle.com/LitlePayPage
   /litle-api2.js
   ```

NOTE: **These URLs should only be used in the testing and certification environment. Do not use this URL in a production environment. Contact your Implementation Consultant for the appropriate production URL.**

3. Submit transactions from your checkout page or mobile application using the **Card Numbers** and **Card Validation Numbers** from Table 2-6. When performing these tests, you can use any expiration date and card type.
4. If testing your Vantiv-Hosted iFrame Integration, use Test Cases 2 through 5.
5. Verify that your results match the **Result** column in Table 2-6.

TABLE 2-6 Expected PayPage Test Results (***NOTE***: Card Numbers below have been split into two parts. Join the two parts to obtain actual number to use.)

Test Case	Card Number	Card Validation Number	Response Code	Result
1	Part 1: 51120100 Part 2: 00000003	Any 3-digit	870 (Success)	PayPage Registration ID is generated and the card is scrubbed before the form is submitted.
2	Part 1: 445701000 Part 2: 00000009	Any 3-digit	871	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (not Mod10).”
3	Part 1: 44570100000 Part 2: 0000000006	Any 3-digit	873	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (too long).”
4	Part 1: 601101 Part 2: 000003	Any 3-digit	872	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (too short).”
5	Part 1: 44570100 Part 2: B00000006	Any 3-digit	874	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (not a number).”
6	Part 1: 60110100 Part 2: 00000003	Any 3-digit	875	Checkout form displays error message to card holder, for example, “We are experiencing technical difficulties. Please try again later or call 555-555-1212.”
7	Part 1: 51234567 Part 2: 898010003	Any 3-digit	876	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (failure from server).”

TABLE 2-6 Expected PayPage Test Results (Continued)(***NOTE***: Card Numbers below have been split into two parts. Join the two parts to obtain actual number to use.)

Test Case	Card Number	Card Validation Number	Response Code	Result
8	Part 1: 3750010 Part 2: 00000005	Any 3-digit	None (Timeout error)	Checkout form displays error message to card holder, for example, “We are experiencing technical difficulties. Please try again later or call 555-555-1212 (timeout).”
9	Part 1: 44570102 Part 2: 00000007	Any 3-digit	889	Checkout form displays error message to card holder, for example, “We are experiencing technical difficulties. Please try again later or call 555-555-1212.”
10	Part 1: 51120100 Part 2: 00000003	abc	881	Checkout form displays error message to card holder, for example “Invalid Card Validation Number - Check and retry (not a number)”.
11	Part 1: 51120100 Part 2: 00000003	12	882	Checkout form displays error message to card holder, for example “Invalid Card Validation Number - Check and retry (too short)”.
12	Part 1: 51120100 Part 2: 00000003	12345	883	Checkout form displays error message to card holder, for example “Invalid Card Validation Number - Check and retry (too long)”.

To test the submission of PayPage data using LitleXML Authorization transactions:

1. Verify that your LitleXML template is coded correctly for this transaction type (see Authorization Transactions on page 38).
2. Submit three Authorization transactions using the PayPage data from Table 2-7.
3. Verify that your `authorizationResponse` values match the **Response Code Expected** column.

TABLE 2-7 PayPage Authorization Transaction Request and Response Data

Test Case	PayPage Registration ID	Exp. Date	Card Validation Number	Response Code Expected
13	(Use the PayPage Registration ID value received when executing Test Case #1)	Any 4-digit	Any 3-digit	000 - Approved

TABLE 2-7 PayPage Authorization Transaction Request and Response Data (Continued)

Test Case	PayPage Registration ID	Exp. Date	Card Validation Number	Response Code Expected
14	cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NWVXE4ZW5UTko4NU9KK3p1L1p1Vzg4YzVPQVlSUHNITG1JN2I0NzlyTg==	1230	987	877 - Invalid PayPage Registration ID
15	RGFQNCt6U1d1M21SeVByVTM4dHlHb1FsVkUrSmpnWXhNY0o5UkMzRlZFanZiUHVnYjN1enJXbG1WSDF4aXlNcA==	1230	987	877 - Expired PayPage Registration ID

A

CODE SAMPLES AND OTHER INFORMATION

This appendix provides code examples and reference material related to integrating the PayPage Solution. The following sections are included:

- HTML Checkout Page Examples
  - HTML Example for Non-PayPage Checkout Page
  - HTML Example for JavaScript API-Integrated Checkout Page
  - HTML Example for Hosted iFrame-Integrated Checkout Page
- Information Sent to Order Processing Systems
- Sample JavaScripts
- LitleXML Elements for PayPage

A.1 HTML Checkout Page Examples

NOTE: This section does not apply to PayPage solutions in a mobile application.

This section provides three HTML checkout page examples:

- HTML Example for Non-PayPage Checkout Page
- HTML Example for JavaScript API-Integrated Checkout Page
- HTML Example for Hosted iFrame-Integrated Checkout Page

A.1.1 HTML Example for Non-PayPage Checkout Page

For comparison purposes, the following HTML sample is for a simple check-out page that is not integrated with PayPage. The check-out form requests the cardholder's name, CVV code, credit card account number, and expiration date.

```
<HTML>
<head>
  <title>Non-PayPage Merchant Checkout</title>
</head>
<BODY>
  <h2>Checkout Form</h2>
  <form method=post id="fCheckout" name="fCheckout"
    action="/merchant101/Merchant101CheckoutServlet">
    <table>
      <tr><td>First Name</td><td><input type="text" id="fName" name="fName" size="20">
</td></tr>
      <tr><td>Last Name</td><td><input type="text" id="lName" name="lName" size="20">
</td></tr>
      <tr><td>Credit Card</td><td><input type="text" id="ccNum" name="ccNum"
size="20"> </td></tr>
      <tr><td>CVV/td><td><input type="text" id="cvv" name="cvv" size="5"> </td></tr>
      <tr><td>Exp Date</td><td><input type="text" id="expDate" name="expDate"
size="5"></td></tr>
      <tr><td> </td><td></tr>
      <tr><td></td><td align="right"><input type="submit"
        value="Check out" id="submitId"/></td></tr>
    </table>
  </form>
</BODY>
</HTML>
```

A.1.2 HTML Example for JavaScript API-Integrated Checkout Page

The HTML code below is an example of a simple checkout page integrated with the JavaScript Customer Browser PayPage solution.

NOTE: The URL in this example (in red) should only be used in the certification and testing environment. Before using your checkout page with PayPage in a production environment, replace the certification URL with the production URL (contact your Implementation Consultant for the appropriate production URL).

```
<HTML>
      <head>
      <title>PayPage Merchant Simple Checkout</title>
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
      type="text/javascript"></script>
      <script src="https://request-prelive.np-securepaypage-litle.com/LitlePayPage/
        litle-api2.js" type="text/javascript"></script>
      <script>
      $(document).ready(
       function(){
         function setLitleResponseFields(response) {
           document.getElementById('response$code').value = response.response;
           document.getElementById('response$message').value = response.message;
           document.getElementById('response$responseTime').value =
      response.responseTime;
           document.getElementById('response$litleTxnId').value = response.litleTxnId;
           document.getElementById('response$type').value = response.type;
           document.getElementById('response$firstSix').value = response.firstSix;
           document.getElementById('response$lastFour').value = response.lastFour;
      }
      function submitAfterLitle (response) {
        setLitleResponseFields(response);
        document.forms['fCheckout'].submit();
      }
      function timeoutOnLitle () {
        alert("We are experiencing technical difficulties.  Please try again later or
call 555-555-1212 (timeout)");
      }
          function onErrorAfterLitle (response) {
            setLitleResponseFields(response);
            if(response.response == '871') {
              alert("Invalid card number.  Check and retry. (Not Mod10)");
            }
            else if(response.response == '872') {
              alert("Invalid card number.  Check and retry. (Too short)");
            }
            else if(response.response == '873') {
              alert("Invalid card number.  Check and retry. (Too long)");
            }
            else if(response.response == '874') {
              alert("Invalid card number.  Check and retry. (Not a number)");
            }
            else if(response.response == '875') {
              alert("We are experiencing technical difficulties. Please try again later
```

Do not use this URL in a production environment. Contact Implementation for the appropriate production URL.

```
        or call 555-555-1212");
              }
              else if(response.response == '876') {
                alert("Invalid card number.  Check and retry. (Failure from Server)");
              }
              else if(response.response == '881') {
                alert("Invalid card validation code.  Check and retry. (Not a number)");
              }
              else if(response.response == '882') {
                alert("Invalid card validation code.  Check and retry. (Too short)");
              }
              else if(response.response == '883') {
                alert("Invalid card validation code.  Check and retry. (Too long)");
            }
            else if(response.response == '889') {
              alert("We are experiencing technical difficulties. Please try again later or
call 555-555-1212");
            }
            return false;
          }
          var formFields = {
            "accountNum"   :document.getElementById('ccNum'),
            "cvv2"   :document.getElementById('cvv2Num'),
"paypageRegistrationId":document.getElementById('response$paypageRegistrationId'),
            "bin"  :document.getElementById('response$bin')
          };
          $("#submitId").click(
            function(){
              // clear test fields
              setLitleResponseFields({"response":"", "message":""});

              var litleRequest = {
                "paypageId" : document.getElementById("request$paypageId").value,
                "reportGroup" : document.getElementById("request$reportGroup").value,
                "orderId" : document.getElementById("request$orderId").value,
                "id" : document.getElementById("request$merchantTxnId").value
                "url" : "https://request-prelive.np-securepaypage-litle.com"
              };
              new LitlePayPage().sendToLitle(litleRequest, formFields, submitAfterLitle,
onErrorAfterLitle, timeoutOnLitle, 5000);
                return false;
            }
          );
        }
      );
  </script>
</head>
<BODY>
  <h2>Checkout Form</h2>
  <form method=post id="fCheckout" name="fCheckout"
action="/merchant101/Merchant101CheckoutServlet">
    <input type="hidden" id="request$paypageId" name="request$paypageId"
value="a2y4o6m8k0"/>
    <input type="hidden" id="request$merchantTxnId" name="request$merchantTxnId"
value="987012"/>
    <input type="hidden" id="request$orderId" name="request$orderId" value="order_123"/>
    <input type="hidden" id="request$reportGroup" name="request$reportGroup"
value="*merchant1500"/>

    <table>
      <tr><td>First Name</td><td><input type="text" id="fName" name="fName"
```

Do not use this URL in a production environment. Contact Implementation for the appropriate production URL.

```
size="20"></td></tr>
      <tr><td>Last Name</td><td><input type="text" id="lName" name="lName"
size="20"></td></tr>
      <tr><td>Credit Card</td><td><input type="text" id="ccNum" name="ccNum"
size="20"></td></tr>
      <tr><td>CVV</td><td><input type="text" id="cvv2num" name="cvv2num"
size="5"></td></tr>
      <tr><td>Exp Date</td><td><input type="text" id="expDate" name="expDate"
size="5"></td></tr>
      <tr><td> </td><td></tr>
      <tr><td></td><td align="right">
      <script>
        document.write('<button type="button" id="submitId" onclick="callLitle()">Check
out with PayPage</button>');
      </script>
      <noscript>
        <button type="button" id="submitId">Enable JavaScript or call us at
555-555-1212</button>
      </noscript>
      </td></tr>
    </table>
    <input type="hidden" id="response$paypageRegistrationId"
name="response$paypageRegistrationId" readOnly="true" value=""/>
    <input type="hidden" id="response$bin" name="response$bin" readOnly="true"/>
    <input type="hidden" id="response$code" name="response$code" readOnly="true"/>
    <input type="hidden" id="response$message" name="response$message" readOnly="true"/>
    <input type="hidden" id="response$responseTime" name="response$responseTime"
readOnly="true"/>
    <input type="hidden" id="response$type" name="response$type" readOnly="true"/>
    <input type="hidden" id="response$litleTxnId" name="response$litleTxnId"
readOnly="true"/>
    <input type="hidden" id="response$firstSix" name="response$firstSix"
readOnly="true"/>
    <input type="hidden" id="response$lastFour" name="response$lastFour"
readOnly="true"/>
  </form>
</BODY>
<script>
/* This is an example of how to handle being unable to download the litle-api2 */
  function callLitle() {
      if(typeof new LitlePayPage() != 'object') {
      alert("We are experiencing technical difficulties.  Please try again later or call
555-555-1212 (API unavailable)" );
  }
}
</script>
</HTML>
```

A.1.3 HTML Example for Hosted iFrame-Integrated Checkout Page

The HTML code below is an example of a simple checkout page integrated with the iFrame API solution.

NOTE: The URL in this example (in red) should only be used in the certification and testing environment. Before using your checkout page with PayPage in a production environment, replace the certification URL with the production URL (contact your Implementation Consultant for the appropriate production URL).

```
<HTML>
<head>
    <title>Merchant1 checkout</title>
    <style>
        body {
            font-size:10pt;
        }
        .checkout {
            background-color:lightgreen;
            width: 50%;
        }
        .testFieldTable {
            background-color:lightgrey;
        }
        #submitId {
            font-weight:bold;
            font-size:12pt;
        }
        form#fCheckout {
        }
    </style>

    <script src="js/jquery.min.js"></script>
    <script src="https://origin-prelive.np-securepaypage-litle.com/LitlePayPage/js/
payframe-client.min.js"></script>
</head>
<body>

    <div class="checkout">
    <h2>Checkout Form</h2>
    <form method=post id="fCheckout" name="fCheckout" onsubmit="return false;">
        <table>
          <tr><td colspan="2">
          <div id="payframe">
          </div>
          </td></tr>
          <tr><td>Paypage Registration ID</td><td><input type="text"
id="paypageRegistrationId" name="paypageRegistrationId" readOnly="true"/>
<--Hidden</td></tr>
          <tr><td>Bin</td><td><input type="text" id="bin" name="bin" readOnly="true"/>
          <--Hidden</td></tr>
          <tr><td></td><td align="right"><input type="submit" id="submitId">Check
out</button></td></tr>
        </table>
```

Do not use this URL in a production environment. Contact Implementation for the appropriate production URL.

```
    </form>
  </div>
      <br/>
  <h3>Test Input Fields</h3>
  <table class="testFieldTable">
      <tr>
          <td>Paypage ID</td><td><input type="text" id="request$paypageId"
name="request$paypageId" value="a2y4o6m8k0" disabled/></td>
          <td>Merchant Txn ID</td><td><input type="text" id="request$merchantTxnId"
name="request$merchantTxnId" value="987012"/></td>
      </tr>
      <tr>
          <td>Order ID</td><td><input type="text" id="request$orderId"
name="request$orderId" value="order_123"/></td>
          <td>Report Group</td><td><input type="text" id="request$reportGroup"
name="request$reportGroup" value="*merchant1500" disabled/></td>
      </tr>
      <tr>
          <td>JS Timeout</td><td><input type="text" id="request$timeout"
name="request$timeout" value="5000" disabled/></td>
      </tr>
    </table>
<h3>Test Output Fields</h3>
<table class="testFieldTable">
    <tr>
        <td>Response Code</td><td><input type="text" id="response$code"
name="response$code" readOnly="true"/></td>
        <td>ResponseTime</td><td><input type="text" id="response$responseTime"
name="response$responseTime" readOnly="true"/></td>
    </tr>
    <tr>
        <td>Response Message</td><td colspan="3"><input type="text"
id="response$message" name="response$message" readOnly="true" size="100"/></td>
    </tr>
    <tr><td> </td><td></tr>
    <tr>
        <td>Vantiv Txn ID</td><td><input type="text" id="response$litleTxnId"
name="response$litleTxnId" readOnly="true"/></td>
        <td>Merchant Txn ID</td><td><input type="text" id="response$merchantTxnId"
name="response$merchantTxnId" readOnly="true"/></td>
    </tr>
    <tr>
        <td>Order ID</td><td><input type="text" id="response$orderId"
name="response$orderId" readOnly="true"/></td>
        <td>Report Group</td><td><input type="text" id="response$reportGroup"
name="response$reportGroup" readOnly="true"/></td>
    </tr>
    <tr><td>Type</td><td><input type="text" id="response$type" name="response$type"
readOnly="true"/></td></tr>
    <tr>
        <td>Expiration Month</td><td><input type="text" id="response$expMonth"
name="response$expMonth" readOnly="true"/></td>
        <td>Expiration Year</td><td><input type="text" id="response$expYear"
name="response$expYear" readOnly="true"/></td>
    </tr>
    <tr><td> </td><td></tr>
    <tr>
        <td>First Six</td><td><input type="text"
id="response$firstSix"name="response$firstSix" readOnly="true"/></td>
      <td>Last Four</td><td><input type="text"
id="response$lastFour"name="response$lastFour" readOnly="true"/></td>
      </tr>
```

```
      <tr><td>Timeout Message</td><td><input type="text" id="timeoutMessage"
name="timeoutMessage" readOnly="true"/></td></tr>
      <tr><td>Expected Results</td>
          <td colspan="3">
          <textarea id="expectedResults" name="expectedResults" rows="5" cols="100"
readOnly="true">
            CC Num           - Token Generated (with simulator)
            4100000000000001 - 1111222233330001
            5123456789012007 - 1112333344442007
            378310203312332  - 111344445552332
            6011000990190005 - 1114555566660005
          </textarea></td>
      </tr>
      <tr>
            <td>Encrypted Card</td>
            <td colspan="3"><textarea id="base64enc" name="base64enc" rows="5"
cols="100" readOnly="true"></textarea></td>
      </tr>
  </table>
<script>

$( document ).ready(function() {
      var startTime;
      var payframeClientCallback = function(response) {
          if (response.timeout) {
              var elapsedTime = new Date().getTime() - startTime;
              document.getElementById('timeoutMessage').value = 'Timed out after ' +
elapsedTime + 'ms';// handle timeout
          }
          else {
              document.getElementById('response$code').value = response.response;
              document.getElementById('response$message').value = response.message;
              document.getElementById('response$responseTime').value =
response.responseTime;
              document.getElementById('response$reportGroup').value =
response.reportGroup;
              document.getElementById('response$merchantTxnId').value = response.id;
              document.getElementById('response$orderId').value = response.orderId;
              document.getElementById('response$litleTxnId').value =
response.litleTxnId;
              document.getElementById('response$type').value = response.type;
              document.getElementById('response$lastFour').value = response.lastFour;
              document.getElementById('response$firstSix').value = response.firstSix;
              document.getElementById('paypageRegistrationId').value =
response.paypageRegistrationId;
              document.getElementById('bin').value = response.bin;
              document.getElementById('response$expMonth').value = response.expMonth;
              document.getElementById('response$expYear').value = response.expYear;
          }
      };

    var configure = {
        "paypageId":document.getElementById("request$paypageId").value,
        "style":"test",
        "height":"200px",
        "reportGroup":document.getElementById("request$reportGroup").value,
        "timeout":document.getElementById("request$timeout").value,
        "div": "payframe",
      "callback": payframeClientCallback,
      "showCvv": true,
      "months": {
          "1":"January",
```

```
            "2":"February",
            "3":"March",
            "4":"April",
            "5":"May",
            "6":"June",
            "7":"July",
            "8":"August",
            "9":"September",
            "10":"October",
            "11":"November",
            "12":"December"
        },
        "numYears": 8,
        "tooltipText": "A CVV is the 3 digit code on the back of your Visa, MasterCard
and Discover or a 4 digit code on the front of your American Express",
        "tabIndex": {
            "cvv":1,
            "accountNumber":2,
            "expMonth":3,
            "expYear":4
        },
        "placeholderText": {
              "cvv":"CVV",
              "accountNumber":"Account Number"
        }
    };
    if(typeof LitlePayframeClient === 'undefined') {
          //This means we couldn't download the payframe-client javascript library
            alert("Couldn't download payframe-client javascript");
    }
    var payframeClient = new LitlePayframeClient(configure);
    document.getElementById("fCheckout").onsubmit = function(){
        var message = {
              "id":document.getElementById("request$merchantTxnId").value,
              "orderId":document.getElementById("request$orderId").value
        };
        startTime = new Date().getTime();
        payframeClient.getPaypageRegistrationId(message);
        return false;
    };
});

</script>
</body>
</HTML>
```

A.2 Information Sent to Order Processing Systems

This section describes the information sent to your order processing system, with and without integrating the PayPage solution.

A.2.1 Information Sent Without Integrating PayPage

If you have already integrated the Vault solution, an XML authorization is submitted with the sensitive card data after your customer completes the checkout form, and a token is stored in its place. The following is an example of the information sent to your order handling system:

cvv - 123
expDate - 1210
fName - Joe
ccNum - <*account number here*>
lName - Buyer

A.2.2 Information Sent with Browser-Based PayPage Integration

When you integrate the PayPage solution, your checkout page stops a transaction when a failure or timeout occurs, thereby not exposing your order processing system to sensitive card data. The success callback stores the response in the hidden form response fields, scrubs the card number, and submits the form. The timeout callback stops the transaction, and the failure callback stops the transaction for non-user errors. In timeout and failure scenarios, nothing is sent to your order handling system.

The following is an example of the information sent to your order handling system on a successful transaction:

cvv - 000
expDate - 1210
fName - Joe
ccNum - xxxxxxxxxxxx0001
lName - Buyer
request$paypageId - a2y4o6m8k0
request$merchantTxnId - 987012
request$orderId - order_123
request$reportGroup - *merchant1500
response$paypageRegistrationId - 1111222233330001
response$bin - 410000
response$code - 870
response$message - Success
response$responseTime - 2010-12-21T12:45:15Z
response$type - VI
response$litleTxnId - 21200000051806

response$firstSix - 410000
response$lastFour - 0001

A.2.3 Information Sent with Mobile API-Based Integration

The following is an example of the information sent to your order handling system on a successful transaction from an application on a mobile device.

paypageId - a2y4o6m8k0
id - 12345
orderId - order_123
reportGroup - *merchant1500
firstSix - 410000
lastFour - 0001

A.3 Sample JavaScripts

This section includes examples of complete PayPage JavaScripts, including:

- Sample PayPage JavaScript (litle-api2.js)
- Sample PayFrame Client (payframe-client.js)
- Sample PayFrame JavaScript (payframe.js)

A.3.1 Sample PayPage JavaScript (litle-api2.js)

The following is an example of a complete PayPage JavaScript with a sample public key (browser-based solutions only).

```
/*!
 * Litle & Co. Pay Page Java Script API Version: 2.1
 * Copyright © 2003-2014, Litle & Co. ALL RIGHTS RESERVED.
 * Includes litle-api2.js
 * http://www.litle.com/
 */

var LitlePayPage = function() {
var LitleEncryption = {
    modulus :
"bc637dd74ba76507dad5af1c7ad6f97dbef5298c3b9f74caea7301347db7b4a8c37f26491863100667246fd45071f334
6bf62239f9b117d06fb67861b66ad0d158beeddd2f6f28be93d846f4c8f9ba1bd7e8f186f36cab0e432f22b3d732c221a
9ff00a9bfacb88b24503e2695fd7237835d4936477b21289478906a49b164f52503c20eb29f11fcbda2af94deb9a0bfde
5a4589276897436315c5d664d60bf10650164f509283aed39747ad5d6cb2bbe54e1b42306e5db37dfd42dcbfcc689e0dd
fe3bc9cb22ae7018e5a4a1ff39813584ac7bd6d6d65ca763f0a672da454081ea20e8b1d403316d80b9353ba396bea8962
b1a7d2f775c76612d857c1f7594f",
    exponent : "10001",
    keyId : "1"
};
return {
        sendToLitle : function(litleRequest, litleFormFields, successCallback,
                errorCallback, timeoutCallback, timeout) {
            var start = new Date().getTime();
            var elapsedTime = 0;
            var litleApiResponse = null;

            setTimeout(
                function(){
                    if (litleApiResponse!=null) {
                        if (litleApiResponse.response == '870') {
                            onSuccess(litleApiResponse);
                            return;
                        }
                        else {
```

```
                onFail(litleApiResponse);
                return;
            }
        }

        elapsedTime = new Date().getTime() - start;
        if(timeout && elapsedTime > timeout) {
            onTimeout();
        }
        else {
            setTimeout(arguments.callee, 10);
        }
    }
    , 10
);

litleFormFields.paypageRegistrationId.value = "";
litleFormFields.bin.value = "";

var accountNumber = litleFormFields.accountNum.value;
accountNumber = removeSpacesHyphens(accountNumber);
var hasCvv2 = (jQuery(litleFormFields.cvv2).length > 0);
if (hasCvv2) {
    var cvv2 = litleFormFields.cvv2.value;
    cvv2 = removeSpacesHyphens(cvv2);
}
var returnCode = isValid(accountNumber);
if(returnCode != "870") {
    return javascriptError(returnCode);
}
if(hasCvv2) {
    returnCode = isValidCvv2(cvv2);
    if(returnCode != "870") {
        return javascriptError(returnCode);
    }
}


var rsa = new RSAKey();
try {
    var validKey = rsa.setPublic(LitleEncryption.modulus, LitleEncryption.exponent);
    var encryptedHex = rsa.encrypt(accountNumber);
    if (hasCvv2)
        var encryptedCvv2Hex = rsa.encrypt(cvv2);
} catch(er) {
    return javascriptError("875");
}

if(encryptedHex) {
    var b64Account = hex2b64(encryptedHex);
    var encAccount = encodeURIComponent(b64Account);

    if (hasCvv2) {
        var b64Cvv2 = hex2b64(encryptedCvv2Hex);
        var encCvv2 = encodeURIComponent(b64Cvv2);
    }
```

```
                try {
                    validateParam("paypageId",litleRequest.paypageId, REQUIRED_VALIDATOR,
REASONABLE_PARAM_LENGTH_VALIDATOR, ALPHANUMERIC_VALIDATOR);
                    validateParam("reportGroup",litleRequest.reportGroup, REQUIRED_VALIDATOR,
REASONABLE_PARAM_LENGTH_VALIDATOR);
                    validateParam("id",litleRequest.id,REQUIRED_VALIDATOR,
REASONABLE_PARAM_LENGTH_VALIDATOR);
                } catch(er) {
                    return javascriptError("889", er);
                }

                var encPaypageId = encodeURIComponent(litleRequest.paypageId);
                var encReportGroup = encodeURIComponent(litleRequest.reportGroup);
                var encOrderId = encodeURIComponent(litleRequest.orderId);
                var encId = encodeURIComponent(litleRequest.id);

                if (hasCvv2) {

jQuery.getJSON(litleRequest.url+"/LitlePayPage/paypage?paypageId="+encPaypageId+"&reportGroup="+e
ncReportGroup+"&id="+encId+"&orderId="+encOrderId+"&encryptedAccount="+encAccount+"&publicKeyId="
+LitleEncryption.keyId+"&encryptedCvv="+encCvv2+"&jsoncallback=?",
                        function (data){
                            litleApiResponse = data;
                        }
                    );
                }
                else {

jQuery.getJSON(litleRequest.url+"/LitlePayPage/paypage?paypageId="+encPaypageId+"&reportGroup="+e
ncReportGroup+"&id="+encId+"&orderId="+encOrderId+"&encryptedAccount="+encAccount+"&publicKeyId="
+LitleEncryption.keyId+"&jsoncallback=?",
                            function (data){
                                litleApiResponse = data;
                            }
                    );
                }
            }
            else {
                return javaScriptError("875");
            }

        function onSuccess(data) {
                litleFormFields.accountNum.value = maskAccountNum(accountNumber);
                var cleanAccountNumber = removeSpacesHyphens(accountNumber);
                data.firstSix = cleanAccountNumber.substring(0,6);
                data.lastFour = cleanAccountNumber.substring(cleanAccountNumber.length-4,
cleanAccountNumber.length);
                if(litleFormFields.extraAccountNums) {
                    for(var key in litleFormFields.extraAccountNums) {
                       var extra = litleFormFields.extraAccountNums[key];
                       extra.value=maskAccountNum(removeSpacesHyphens(extra.value));
                }
        }
        if (hasCvv2)
                litleFormFields.cvv2.value="000";
```

```
        litleFormFields.paypageRegistrationId.value = data.paypageRegistrationId;
        litleFormFields.bin.value = data.bin;
        successCallback(data);
}


            function onFail(data) {
                errorCallback(data);
            }

            function onTimeout() {
                timeoutCallback();
            }

            function maskAccountNum(accountNumber) {
                if(!accountNumber) {
                    return accountNumber;
                }
                accountNumber =
accountNumber.substring(0,accountNumber.length-4).replace(/./g,"X").concat(accountNumber.substrin
g(accountNumber.length-4));
                return accountNumber;
            }

            function isMod10(num) {
                num = (num + '').split('').reverse();
                if (!num.length)
                    return false;
                var total = 0, i;
                for (i = 0; i < num.length; i++) {
                    num[i] = parseInt(num[i])
                    total += i % 2 ? 2 * num[i] - (num[i] > 4 ? 9 : 0) : num[i];
                }
                return (total % 10) == 0;
            }

            function isValid(accountNumber) {
                if(accountNumber.length < 13) {
                    return "872";
                }
                else if(accountNumber.length > 19) {
                    return "873";
                }
                else if(!accountNumber.match(/^[0-9]{13,19}$/)) {
                    return "874";
                }
                else if(!isMod10(accountNumber)) {
                    return "871";
                }
                else {
                    return "870";
                }
            }

            function isValidCvv2(cvv2) {
```

```
            if(cvv2.length < 3) {
                return "882";
            }
            else if(cvv2.length > 4) {
                return "883";
            }
            else if(!cvv2.match(/^\d\d\d\d?$/)) {
                return "881";
            }
            else {
                return "870";
            }
        }

        function validateParam() {
            var paramName = arguments[0];
            var paramValue = arguments[1];
            for(var i = 2; i < arguments.length; i++) {
                arguments[i](paramName,paramValue);
            }
        }

        function REQUIRED_VALIDATOR(paramName,paramValue) {
            if(paramValue.length == 0) {
                throw "Parameter " + paramName + " is required";
            }
        }

        function REASONABLE_PARAM_LENGTH_VALIDATOR(paramName,paramValue) {
            if(paramValue.length > 1024) {
              throw "Parameter " + paramName + " is too long.  Length is " + paramValue.length;
            }
        }

        function ALPHANUMERIC_VALIDATOR(paramName,paramValue) {
            if(!paramValue.match(/^[0-9a-zA-Z]+$/)) {
                throw "Parameter " + paramName + " with value " + paramValue + " is not
alphanumeric";
            }
        }

        function javascriptError(code, message) {
            var jsError = {
                "response" : null,
                "message" : null
            };

            var returnCodeMap = {
                "870" : "Success",
                "871" : "Account number not mod10",
                "872" : "Account number too short",
                "873" : "Account number too long",
                "874" : "Account number not numeric",
                "875" : "Unable to encrypt field",
                "876" : "Account number invalid",
                "881" : "Card validation num not numeric",
```

```
                    "882" : "Card validation num too short",
                    "883" : "Card validation num too long",
                    "889" : "Failure"
                };

                function padzero(n) {
                    return n < 10 ? '0' + n : n;
                }
                function toISOString(d) {
                    return d.getUTCFullYear() + '-' +  padzero(d.getUTCMonth() + 1) + '-' +
padzero(d.getUTCDate()) + 'T' + padzero(d.getUTCHours()) + ':' +  padzero(d.getUTCMinutes()) + ':'
+ padzero(d.getUTCSeconds());
                }

                jsError.response = code;
                if(message == undefined) {
                    jsError.message = returnCodeMap[code];
                }
                else {
                    jsError.message = message;
                }
                jsError.responseTime = toISOString(new Date());
                jsError.reportGroup = litleRequest.reportGroup;
                jsError.id = litleRequest.id;
                jsError.orderId = litleRequest.orderId;
                litleApiResponse = jsError;
            }

            function removeSpacesHyphens(txt) {
                return txt.replace(/[ -]/gi,"");
            }
        }
    };
};
```

A.3.2 Sample PayFrame Client (payframe-client.js)

The following is an example of a complete PayFrame Client JavaScript (for Vantiv-Hosted iFrame solutions only). Your page interacts with with this JavaScript, which runs inside the iFrame.

```
/* global jQuery */
/*!
 * Litle & Co. Pay Frame Java Script API Version: 1.0 Copyright © 2003-2015,
 * Litle & Co. ALL RIGHTS RESERVED. Includes payframe-client.js https://www.litle.com
 */
jQuery(document).ready(function () {
    console.log("payframe-client.js: ready");
});
var configFromMerchant;

var eventHandler = function (e) {
    "use strict";
```

```
    var response = JSON.parse(e.data);
    console.log(response);

    if (response.ready) {
        var payframe = document.getElementById("vantiv-payframe").contentWindow;
        configFromMerchant.action="configure";
        payframe.postMessage(JSON.stringify(configFromMerchant), "*");
    }
    else if (response.height) {
        var iframe = document.getElementById("vantiv-payframe");
        iframe.height = response.height;
    }
    else {
        configFromMerchant.callback(response);
    }
};

if (window.addEventListener) {
    window.addEventListener("message", eventHandler, false);
} else if (window.attachEvent) {
    window.attachEvent("onmessage", eventHandler);
}

var LitlePayframeClient = function (configuration) {
    "use strict";

    configFromMerchant = configuration;
    var div = configFromMerchant.div;
    var paypageId = configFromMerchant.paypageId;
    var style = configFromMerchant.style;
    var height = configFromMerchant.height;
    jQuery("#"+div).html("<iframe id='vantiv-payframe' name='vantiv-payframe' frameborder='0'
src='/LitlePayPage/payframe_"+paypageId+"_"+style+".html' height='"+height+"'
width='100%'></iframe>");

    return {
        getPaypageRegistrationId : function (message) {
            console.log("payframe-client.js: getPaypageRegistrationId");
            console.log(message);
            message.action="getPaypageRegistrationId";
            message.paypageId = configFromMerchant.paypageId;
            message.reportGroup = configFromMerchant.reportGroup;
            message.timeout = configFromMerchant.timeout;
            var payframe = document.getElementById("vantiv-payframe").contentWindow;
            payframe.postMessage(JSON.stringify(message), "*");
        },

        autoAdjustHeight : function () {
            console.log("payframe-client.js: autoAdjustHeight");
            var message = {"action":"getDocHeight"};
            var payframe = document.getElementById("vantiv-payframe").contentWindow;
            payframe.postMessage(JSON.stringify(message), "*");
        }
    };
};
```

```
/*!json2.js*/
if(typeof JSON!=="object"){JSON={}}(function(){function f(n){return n<10?"0"+n:n}if(typeof
Date.prototype.toJSON!=="function"){Date.prototype.toJSON=function(){return
isFinite(this.valueOf())?this.getUTCFullYear()+"-"+f(this.getUTCMonth()+1)+"-"+f(this.getUTCDate(
))+"T"+f(this.getUTCHours())+":"+f(this.getUTCMinutes())+":"+f(this.getUTCSeconds())+"Z":null};St
ring.prototype.toJSON=Number.prototype.toJSON=Boolean.prototype.toJSON=function(){return
this.valueOf()}}var cx,escapable,gap,indent,meta,rep;function
quote(string){escapable.lastIndex=0;return
escapable.test(string)?'"'+string.replace(escapable,function(a){var c=meta[a];return typeof
c==="string"?c:"\\u"+("0000"+a.charCodeAt(0).toString(16)).slice(-4)})+'"':'"'+string+'"'}functio
n str(key,holder){var i,k,v,length,mind=gap,partial,value=holder[key];if(value&&typeof
value==="object"&&typeof value.toJSON==="function"){value=value.toJSON(key)}if(typeof
rep==="function"){value=rep.call(holder,key,value)}switch(typeof value){case"string":return
quote(value);case"number":return
isFinite(value)?String(value):"null";case"boolean":case"null":return
String(value);case"object":if(!value){return"null"}gap+=indent;partial=[];if(Object.prototype.toS
tring.apply(value)==="[object
Array]"){length=value.length;for(i=0;i<length;i+=1){partial[i]=str(i,value)||"null"}v=partial.len
gth===0?"[]":gap?"[\n"+gap+partial.join(",\n"+gap)+"\n"+mind+"]":"["+partial.join(",")+"]";gap=mi
nd;return v}if(rep&&typeof rep==="object"){length=rep.length;for(i=0;i<length;i+=1){if(typeof
rep[i]==="string"){k=rep[i];v=str(k,value);if(v){partial.push(quote(k)+(gap?":
":":")+v)}}}}else{for(k in
value){if(Object.prototype.hasOwnProperty.call(value,k)){v=str(k,value);if(v){partial.push(quote(
k)+(gap?":
":":")+v)}}}}v=partial.length===0?"{}":gap?"{\n"+gap+partial.join(",\n"+gap)+"\n"+mind+"}":"{"+pa
rtial.join(",")+"}";gap=mind;return v}}if(typeof
JSON.stringify!=="function"){escapable=/[\\\"\x00-\x1f\x7f-\x9f\u00ad\u0600-\u0604\u070f\u17b4\u1
7b5\u200c-\u200f\u2028-\u202f\u2060-\u206f\ufeff\ufff0-\uffff]/g;meta={"\b":"\\b","\t":"\\t","\n"
:"\\n","\f":"\\f","\r":"\\r",'"':'\\"',"\\":"\\\\"};JSON.stringify=function(value,replacer,space)
{var i;gap="";indent="";if(typeof space==="number"){for(i=0;i<space;i+=1){indent+="
"}}else{if(typeof space==="string"){indent=space}}rep=replacer;if(replacer&&typeof
replacer!=="function"&&(typeof replacer!=="object"||typeof replacer.length!=="number")){throw new
Error("JSON.stringify")}return str("",{"":value})}}if(typeof
JSON.parse!=="function"){cx=/[\u0000\u00ad\u0600-\u0604\u070f\u17b4\u17b5\u200c-\u200f\u2028-\u20
2f\u2060-\u206f\ufeff\ufff0-\uffff]/g;JSON.parse=function(text,reviver){var j;function
walk(holder,key){var k,v,value=holder[key];if(value&&typeof value==="object"){for(k in
value){if(Object.prototype.hasOwnProperty.call(value,k)){v=walk(value,k);if(v!==undefined){value[
k]=v}else{delete value[k]}}}}return
reviver.call(holder,key,value)}text=String(text);cx.lastIndex=0;if(cx.test(text)){text=text.repla
ce(cx,function(a){return"\\u"+("0000"+a.charCodeAt(0).toString(16)).slice(-4)})}if(/^[\],:{}\s]*$
/.test(text.replace(/\\(?:["\\\/bfnrt]|u[0-9a-fA-F]{4})/g,"@").replace(/"[^"\\\n\r]*"|true|false|
null|-?\d+(?:\.\d*)?(?:[eE][+\-]?\d+)?/g,"]").replace(/(?:^|:|,)(?:\s*\[)+/g,""))){j=eval("("+tex
t+")");return typeof reviver==="function"?walk({"":j},""):j}throw new
SyntaxError("JSON.parse")}}}());
```

A.3.3 Sample PayFrame JavaScript (payframe.js)

The following is an example of a complete PayFrame JavaScript (for Vantiv-Hosted iFrame solutions only). The iFrame Client interacts with this JavaScript, which runs inside of the iFrame.

```
/* global jQuery */
/* global LitlePayPage */
/* global yepnope */
/*!
 * Litle & Co. Pay Frame Java Script API Version: 1.0 Copyright Â© 2003-2015,
 * Litle & Co. ALL RIGHTS RESERVED. Includes payframe.js https://www.litle.com
 */
$( document ).ready(function() {
    console.log("payframe.js: ready");
    //Remove error handling classes on focus
    $("#accountNumber").focus(function() {
        $(".numberDiv").removeClass("error error-871 error-872error-874 error-876");
    });
    $("#cvv").focus(function() {
        $(".cvvDiv").removeClass("error error-881 error-882");
    });
    var response = {ready : true };
    yepnope({
        test : window.JSON,
        nope : [ '/LitlePayPage/js/json2-20140404.min.js' ],
        complete : function () {
            window.parent.postMessage(JSON.stringify(response), '*');
        }
    });
});
var LitlePayFrame = function () {
    "use strict";
    var shouldUseCvv = false;

    var submitAfterLitle = function (response) {
        console.log("payframe.js: submitAfterLitle");
        response.expMonth = jQuery("#expMonth").val();
        response.expYear = jQuery("#expYear").val();
        window.parent.postMessage(JSON.stringify(response), '*');
    };

    var onErrorAfterLitle = function (response) {
        console.log("payframe.js: onErrorAfterLitle");
        var errorCode = response.response;
        /*
         "871" : "Account number not mod10",
         "872" : "Account number too short",
         "873" : "Account number too long",
         "874" : "Account number not numeric",
         "875" : "Unable to encrypt field",
         "876" : "Account number invalid",
         "881" : "Card validation num not numeric",
         "882" : "Card validation num too short",
         "883" : "Card validation num too long",
         "889" : "Failure"
         */
```

```
    //we only check for user error here, so no error classes are added for 875 or 889
    if(errorCode === "871") {
        $(".numberDiv").addClass("error");
        $(".numberDiv").addClass("error-871");
    }
    else if(errorCode === "872") {
        $(".numberDiv").addClass("error");
        $(".numberDiv").addClass("error-872");
    }
    else if(errorCode === "874") {
        $(".numberDiv").addClass("error");
        $(".numberDiv").addClass("error-874");
    }
    else if(errorCode === "876") {
        $(".numberDiv").addClass("error");
        $(".numberDiv").addClass("error-876");
    }
    else if(errorCode === "881") {
        $(".cvvDiv").addClass("error");
        $(".cvvDiv").addClass("error-881");
    }
    else if(errorCode === "882") {
        $(".cvvDiv").addClass("error");
        $(".cvvDiv").addClass("error-882");
    }
    //all errors (user or not) are returned to the outer frame
    window.parent.postMessage(JSON.stringify(response), '*');
}

var onTimeoutAfterLitle = function () {
    var response = {timeout : true };
   console.log("payframe.js: onTimeoutAfterLitle");
    window.parent.postMessage(JSON.stringify(response), '*');
};

var getPaypageRegistrationId = function (message) {
    console.log("payframe.js: getPaypageRegistrationId");
    console.log(message);
    var litleRequest = {
        "paypageId" : message.paypageId,
        "reportGroup" : message.reportGroup,
        "orderId" : message.orderId,
        "id" : message.id,
        "url" : "" // TODO Different in non-dev?
    };

    if (message.pciNonSensitive){
    litleRequest.pciNonSensitive = true;
    }

    var formFields = {
        "accountNum" : document.getElementById('accountNumber'),
        "paypageRegistrationId" : document
                .getElementById('paypageRegistrationId'),
        "bin" : document.getElementById('bin')
    };
```

```
    if (shouldUseCvv) {
        formFields.cvv2 = document.getElementById('cvv');
    }

    var timeout = message.timeout;
    var litlePaypage = new LitlePayPage();
    litlePaypage.sendToLitle(litleRequest, formFields, submitAfterLitle,
            onErrorAfterLitle, onTimeoutAfterLitle, timeout);
};

var showCvv = function (message) {
    console.log("payframe.js: showCvv");
    console.log(message);
    if (message) {
        jQuery(".cvvDiv").show();
        shouldUseCvv = true;
    } else {
        jQuery(".cvvDiv").hide();
        shouldUseCvv = false;
    }
};

var setMonths = function (message) {
    console.log("payframe.js: setMonths");
    console.log(message);
    var months = message;
    jQuery("#expMonth")[0][0].text = months['1'];
    jQuery("#expMonth")[0][1].text = months['2'];
    jQuery("#expMonth")[0][2].text = months['3'];
    jQuery("#expMonth")[0][3].text = months['4'];
    jQuery("#expMonth")[0][4].text = months['5'];
    jQuery("#expMonth")[0][5].text = months['6'];
    jQuery("#expMonth")[0][6].text = months['7'];
    jQuery("#expMonth")[0][7].text = months['8'];
    jQuery("#expMonth")[0][8].text = months['9'];
    jQuery("#expMonth")[0][9].text = months['10'];
    jQuery("#expMonth")[0][10].text = months['11'];
    jQuery("#expMonth")[0][11].text = months['12'];
};

var setYears = function (message) {
    console.log("payframe.js: setYears");
    console.log(message);
    var numYearsToAdd = message;

    var numOfYearsToRemove = jQuery("#expYear option").size();
    for ( var i = 0; i < numOfYearsToRemove; i++) {
        jQuery("#expYear option:first").remove();
    }

    var currentYear = new Date().getFullYear();
    for ( var j = 0; j < numYearsToAdd; j++) {
        var o = new Option(currentYear + j, currentYear + j - 2000);
        jQuery(o).html(currentYear + j); // For IE8
        jQuery("#expYear").append(o);
    }
```

```
};

var setTooltipText = function (message) {
    console.log("payframe.js: setTooltipText");
    var text = message;
    jQuery(".tooltip").attr("title", text);
};

var setTabIndex = function (message) {
    console.log("payframe.js: setTabIndex");
    console.log(message);
    if(message.cvv) {
        $("#cvv").attr("tabindex",message.cvv);
    }
    if(message.accountNumber) {
        $("#accountNumber").attr("tabindex",message.accountNumber);
    }
    if(message.expMonth) {
        $("#expMonth").attr("tabindex",message.expMonth);
    }
    if(message.expYear) {
        $("#expYear").attr("tabindex",message.expYear);
    }
};

var setPlaceholderText = function(message) {
    console.log("payframe.js: setPlaceholderText");
    console.log(message);
    if(message.cvv) {
        $("#cvv").attr("placeholder",message.cvv);
    }
    if(message.accountNumber) {
        $("#accountNumber").attr("placeholder", message.accountNumber);
    }
}

var getDocHeight = function() {
    var D = document;
    var height = Math.max(
        Math.max(D.body.scrollHeight, D.documentElement.scrollHeight),
        Math.max(D.body.offsetHeight, D.documentElement.offsetHeight),
        Math.max(D.body.clientHeight, D.documentElement.clientHeight)
    );
    var response = {"height" : height };
    yepnope({
        test : window.JSON,
        nope : [ '/LitlePayPage/js/json2-20140404.min.js' ],
        complete : function () {
            window.parent.postMessage(JSON.stringify(response), '*');
        }
    });
}

return {
    receivedMessageFromOuterFrame : function (message) {
        console.log("payframe.js: receivedMessageFromOuterFrame");
```

```
            console.log(message);
            var action = message.action;
            console.log("payframe.js: action: " + action);
            if (action === "getPaypageRegistrationId") {
                getPaypageRegistrationId(message);
            } else if (action === "configure") {
                if (message.months) {
                    setMonths(message.months);
                }
                if (message.numYears) {
                    setYears(message.numYears);
                }
                if (message.showCvv) {
                    showCvv(message.showCvv);
                }
                if (message.tooltipText) {
                    setTooltipText(message.tooltipText);
                }
                if (message.tabIndex) {
                    setTabIndex(message.tabIndex)
                }
                if (message.placeholderText) {
                    setPlaceholderText(message.placeholderText);
                }
            } else if (action === "getDocHeight") {
                getDocHeight();
            } else {
                console.log("payframe.js: received unknown action: " + event.data.action);
            }

        }
    };
};

var litlePayFrame = new LitlePayFrame();

var eventHandler = function (event) {
    console.log(event.data);
    var messageAsString = event.data;
    yepnope({
        test : window.JSON,
        nope : [ '/LitlePayPage/js/json2-20140404.min.js' ],
        complete : function () {
            var message = JSON.parse(messageAsString);
            litlePayFrame.receivedMessageFromOuterFrame(message);
        }
    });
};

if (window.addEventListener) {
    window.addEventListener("message", eventHandler, false);
} else if (window.attachEvent) {
    window.attachEvent("onmessage", eventHandler);
}
```

A.4 LitleXML Elements for PayPage

This section provides definitions for the elements used in the LitleXML for PayPage transactions.

Use this information in combination with the various LitleXML schema files to assist you as you build the code necessary to submit PayPage 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 LitleXML requests and responses using these elements, as well as XML examples, see LitleXML Transaction Examples When Using PayPage on page 37. For a comprehensive list of all LitleXML elements and usage, see Chapter 4, "LitleXML Elements" in the *Vantiv LitleXML Reference Guide*.

The XML elements defined in this section are as follows (listed alphabetically):

- cardValidationNum
- expDate
- paypage
- paypageRegistrationId
- registerTokenRequest
- registerTokenResponse

A.4.1 cardValidationNum

The `<cardValidationNum>` element is an optional child of the `<card>`, `<paypage>`, `<token>`, `<registerTokenRequest>`, or `<updateCardValidatinNumOnToken>` 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`, our platform encrypts and stores the value on a temporary basis for later use in a tokenized Authorization or 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

A.4.2 expDate

The `<expDate>` element is a child of the `<card>`, `<paypage>`, `<token>`, or other element listed below, which specifies the expiration date of the card and is required for card-not-present transactions.

NOTE: Although the schema defines the `<expDate>` element as an *optional* child of the `<card>`, `<token>` and `<paypage>` elements, you must submit a value for card-not-present transactions.

Type = String; **minLength** = 4; **maxLength** = 4

Parent Elements:

card, newCardInfo, newCardTokenInfo, originalCard, originalCardInfo, originalCardTokenInfo, originalToken, paypage, token, updatedCard, updatedToken

Attributes:

None

Child Elements:

None

NOTE: You should submit whatever expiration date you have on file, regardless of whether or not it is expired/stale.

We recommend all merchant with recurring and/or installment payments participate in the Automatic Account Updater program.

A.4.3 paypage

The <paypage> element defines PayPage account information. It replaces the <card> or <token> elements in transactions using the PayPage feature of the Vault solution.

Parent Elements:

authorization, sale, captureGivenAuth, forceCapture, credit, updateSubscription

Attributes:

None

Child Elements:

Required: paypageRegistrationId

Optional: cardValidationNum, expDate, type

> **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.**

Example: paypage Structure

```
<paypage>
  <paypageRegistrationId>Registration ID from PayPage</paypageRegistrationId>
  <expDate>Expiration Date</expDate>
  <cardValidationNum>Card Validation Number</cardValidationNum>
  <type>Method of Payment</type>
</paypage>
```

A.4.4 paypageRegistrationId

The `<paypageRegistrationId>` element is a child of the `<paypage>` element and the `<registerTokenRequest>` element. It replaces the `<accountNumber>` or `<echeckForToken>` elements in a Register Token transaction. It specifies the PayPage Registration ID generated by securepaypage-litle.com (PayPage). It can also be used in a Register Token Request to obtain a token, based on PayPage 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

A.4.5 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, eCheck account number, or PayPage Registration Id 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:

litleOnlineRequest, batchRequest

Attributes:

Attribute Name	Type	Required?	Description
id	String	No	A unique identifier assigned by the presenter and mirrored back in the response. **minLength** = N/A **maxLength** = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. **minLength** = N/A **maxLength** = 50
reportGroup	String	Yes	Required attribute defining the merchant sub-group in iQ where this transaction displays. **minLength** = 1 **maxLength** = 25

Child Elements:

Required: (choice of) accountNumber, echeckForToken, mpos, or paypageRegistrationId

Optional: orderId, cardValidationNum

NOTE: The use of the `<cardValidationNum>` element in the `<registertokenRequest>` only applies when you submit an `<accountNumber>` element.

A.4.6 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, eCheck account number, or PayPage registration ID for tokenization in a Register Token transaction.

Parent Elements:

litleOnlineResponse, batchResponse

Attributes:

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

Child Elements:

Required: litleTxnId, response, message, responseTime

Optional: eCheckAccountSuffix, orderId, litleToken, bin, type

B

CSS PROPERTIES FOR IFRAME API

This appendix provides a list of Cascading Style Sheet (CSS) properties, for use when creating your iFrame implementation of PayPage, as listed in the CSS specification V1-3.

See the section, Creating a Customized CSS for Vantiv-Hosted iFrame on page 10 before using the properties listed here.

Except as marked (shaded items), the properties listed in the tables below are allowable when styling your CSS for PayPage iFrame. Allowable values have been 'white-listed' programmatically. See Table B-23, "CSS Properties Excluded From the White List (not allowed)" for more information.

B.1 CSS Property Groups

- Color Properties
- Background and Border Properties
- Basic Box Properties
- Flexible Box Layout
- Text Properties
- Text Decoration Properties
- Font Properties
- Writing Modes Properties
- Table Properties
- Lists and Counters Properties
- Animation Properties
- Transform Properties
- Transitions Properties
- Basic User Interface Properties
- Multi-Column Layout Properties
- Paged Media
- Generated Content for Paged Media
- Filter Effects Properties
- Image Values and Replaced Content
- Masking Properties
- Speech Properties
- Marquee Properties

TABLE B-1 Color Properties

Property	Description
color	Sets the color of text
opacity	Sets the opacity level for an element

TABLE B-2 Background and Border Properties

Property	Description
background **(Do not use)**	*Sets all the background properties in one declaration*
background-attachment **(Do not use)**	*Sets whether a background image is fixed or scrolls with the rest of the page*
background-color	Sets the background color of an element
background-image	Sets the background image for an element
background-position **(Do not use)**	*Sets the starting position of a background image*
background-repeat **(Do not use)**	*Sets how a background image will be repeated*
background-clip	Specifies the painting area of the background
background-origin **(Do not use)**	*Specifies the positioning area of the background images*
background-size **(Do not use)**	*Specifies the size of the background images*
border	Sets all the border properties in one declaration
border-bottom	Sets all the bottom border properties in one declaration
border-bottom-color	Sets the color of the bottom border
border-bottom-left-radius	Defines the shape of the border of the bottom-left corner
border-bottom-right-radius	Defines the shape of the border of the bottom-right corner
border-bottom-style	Sets the style of the bottom border
border-bottom-width	Sets the width of the bottom border
border-color	Sets the color of the four borders
border-image **(Do not use)**	*A shorthand property for setting all the border-image-* properties*

TABLE B-2 Background and Border Properties (Continued)

Property	Description
border-image-outset **(Do not use)**	*Specifies the amount by which the border image area extends beyond the border box*
border-image-repeat **(Do not use)**	*Specifies whether the image-border should be repeated, rounded or stretched*
border-image-slice	Specifies the inward offsets of the image-border
border-image-source **(Do not use)**	*Specifies an image to be used as a border*
border-image-width **(Do not use)**	*Specifies the widths of the image-border*
border-left	Sets all the left border properties in one declaration
border-left-color	Sets the color of the left border
border-left-style	Sets the style of the left border
border-left-width	Sets the width of the left border
border-radius	A shorthand property for setting all the four border-*-radius properties
border-right	Sets all the right border properties in one declaration
border-right-color	Sets the color of the right border
border-right-style	Sets the style of the right border
border-right-width	Sets the width of the right border
border-style	Sets the style of the four borders
border-top	Sets all the top border properties in one declaration
border-top-color	Sets the color of the top border
border-top-left-radius	Defines the shape of the border of the top-left corner
border-top-right-radius	Defines the shape of the border of the top-right corner
border-top-style	Sets the style of the top border
border-top-width	Sets the width of the top border
border-width	Sets the width of the four borders
box-decoration-break	Sets the behavior of the background and border of an element at page-break, or, for in-line elements, at line-break.
box-shadow	Attaches one or more drop-shadows to the box

TABLE B-3 Basic Box Properties

Property	Description
bottom	Specifies the bottom position of a positioned element
clear	Specifies which sides of an element where other floating elements are not allowed
clip	Clips an absolutely positioned element
display	Specifies how a certain HTML element should be displayed
float	Specifies whether or not a box should float
height	Sets the height of an element
left	Specifies the left position of a positioned element
overflow	Specifies what happens if content overflows an element's box
overflow-x	Specifies whether or not to clip the left/right edges of the content, if it overflows the element's content area
overflow-y	Specifies whether or not to clip the top/bottom edges of the content, if it overflows the element's content area
padding	Sets all the padding properties in one declaration
padding-bottom	Sets the bottom padding of an element
padding-left	Sets the left padding of an element
padding-right	Sets the right padding of an element
padding-top	Sets the top padding of an element
position	Specifies the type of positioning method used for an element (static, relative, absolute or fixed)
right	Specifies the right position of a positioned element
top	Specifies the top position of a positioned element
visibility	Specifies whether or not an element is visible
width	Sets the width of an element
vertical-align	Sets the vertical alignment of an element
z-index	Sets the stack order of a positioned element

TABLE B-4 Flexible Box Layout

Property	Description
align-content	Specifies the alignment between the lines inside a flexible container when the items do not use all available space.
align-items	Specifies the alignment for items inside a flexible container.
align-self	Specifies the alignment for selected items inside a flexible container.
display	Specifies how a certain HTML element should be displayed
flex	Specifies the length of the item, relative to the rest
flex-basis	Specifies the initial length of a flexible item
flex-direction	Specifies the direction of the flexible items
flex-flow	A shorthand property for the flex-direction and the flex-wrap properties
flex-grow	Specifies how much the item will grow relative to the rest
flex-shrink	Specifies how the item will shrink relative to the rest
flex-wrap	Specifies whether the flexible items should wrap or not
justify-content	Specifies the alignment between the items inside a flexible container when the items do not use all available space.
margin	Sets all the margin properties in one declaration
margin-bottom	Sets the bottom margin of an element
margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element
max-height	Sets the maximum height of an element
max-width	Sets the maximum width of an element
min-height	Sets the minimum height of an element
min-width	Sets the minimum width of an element
order	Sets the order of the flexible item, relative to the rest

TABLE B-5 Text Properties

Property	Description
hanging-punctuation	Specifies whether a punctuation character may be placed outside the line box
hyphens	Sets how to split words to improve the layout of paragraphs
letter-spacing	Increases or decreases the space between characters in a text
line-break	Specifies how/if to break lines
line-height	Sets the line height
overflow-wrap	Specifies whether or not the browser may break lines within words in order to prevent overflow (when a string is too long to fit its containing box)
tab-size	Specifies the length of the tab-character
text-align	Specifies the horizontal alignment of text
text-align-last	Describes how the last line of a block or a line right before a forced line break is aligned when text-align is “justify”
text-combine-upright	Specifies the combination of multiple characters into the space of a single character
text-indent	Specifies the indentation of the first line in a text-block
text-justify	Specifies the justification method used when text-align is “justify”
text-transform	Controls the capitalization of text
white-space	Specifies how white-space inside an element is handled
word-break	Specifies line breaking rules for non-CJK scripts
word-spacing	Increases or decreases the space between words in a text
word-wrap	Allows long, unbreakable words to be broken and wrap to the next line

TABLE B-6 Text Decoration Properties

Property	Description
text-decoration	Specifies the decoration added to text
text-decoration-color	Specifies the color of the text-decoration
text-decoration-line	Specifies the type of line in a text-decoration

TABLE B-6 Text Decoration Properties (Continued)

Property	Description
text-decoration-style	Specifies the style of the line in a text decoration
text-shadow	Adds shadow to text
text-underline-position	Specifies the position of the underline which is set using the text-decoration property

TABLE B-7 Font Properties

Property	Description
@font-face **(Do not use)**	*A rule that allows websites to download and use fonts other than the "web-safe" fonts*
@font-feature-values	Allows authors to use a common name in font-variant-alternate for feature activated differently in OpenType
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-feature-settings	Allows control over advanced typographic features in OpenType fonts
font-kerning	Controls the usage of the kerning information (how letters are spaced)
font-language-override	Controls the usage of language-specific glyphs in a typeface
font-size	Specifies the font size of text
font-size-adjust	Preserves the readability of text when font fallback occurs
font-stretch	Selects a normal, condensed, or expanded face from a font family
font-style	Specifies the font style for text
font-synthesis	Controls which missing typefaces (bold or italic) may be synthesized by the browser
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-variant-alternates	Controls the usage of alternate glyphs associated to alternative names defined in @font-feature-values
font-variant-caps	Controls the usage of alternate glyphs for capital letters
font-variant-east-asian	Controls the usage of alternate glyphs for East Asian scripts (e.g Japanese and Chinese)

TABLE B-7 Font Properties (Continued)

Property	Description
font-variant-ligatures	Controls which ligatures and contextual forms are used in textual content of the elements it applies to
font-variant-numeric	Controls the usage of alternate glyphs for numbers, fractions, and ordinal markers
font-variant-position	Controls the usage of alternate glyphs of smaller size positioned as superscript or subscript regarding the baseline of the font
font-weight	Specifies the weight of a font

TABLE B-8 Writing Modes Properties

Property	Description
direction	Specifies the text direction/writing direction
text-orientation	Defines the orientation of the text in a line
text-combine-upright	Specifies the combination of multiple characters into the space of a single character
unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
writing-mode	

TABLE B-9 Table Properties

Property	Description
border-collapse	Specifies whether or not table borders should be collapsed
border-spacing	Specifies the distance between the borders of adjacent cells
caption-side	Specifies the placement of a table caption
empty-cells	Specifies whether or not to display borders and background on empty cells in a table
table-layout	Sets the layout algorithm to be used for a table

TABLE B-10 Lists and Counters Properties

Property	Description
counter-increment	Increments one or more counters
counter-reset	Creates or resets one or more counters
list-style	Sets all the properties for a list in one declaration
list-style-image ***(Do not use)***	*Specifies an image as the list-item marker*
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow
list-style-type	Specifies the type of list-item marker

TABLE B-11 Animation Properties

Property	Description
@keyframes	Specifies the animation
animation	A shorthand property for all the animation properties below, except the animation-play-state property
animation-delay	Specifies when the animation will start
animation-direction	Specifies whether or not the animation should play in reverse on alternate cycles
animation-duration	Specifies how many seconds or milliseconds an animation takes to complete one cycle
animation-fill-mode	Specifies what values are applied by the animation outside the time it is executing
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies a name for the @keyframes animation
animation-timing-function	Specifies the speed curve of the animation
animation-play-state	Specifies whether the animation is running or paused

TABLE B-12 Transform Properties

Property	Description
backface-visibility	Defines whether or not an element should be visible when not facing the screen

TABLE B-12 Transform Properties (Continued)

Property	Description
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space

TABLE B-13 Transitions Properties

Property	Description
transition	A shorthand property for setting the four transition properties
transition-property	Specifies the name of the CSS property the transition effect is for
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-timing-function	Specifies the speed curve of the transition effect
transition-delay	Specifies when the transition effect will start

TABLE B-14 Basic User Interface Properties

Property	Description
box-sizing	Tells the browser what the sizing properties (width and height) should include
content	Used with the :before and :after pseudo-elements, to insert generated content
cursor **(Do not use)**	*Specifies the type of cursor to be displayed*
icon **(Do not use)**	*Provides the author the ability to style an element with an iconic equivalent*
ime-mode	Controls the state of the input method editor for text fields
nav-down	Specifies where to navigate when using the arrow-down navigation key

TABLE B-14 Basic User Interface Properties (Continued)

Property	Description
nav-index	Specifies the tabbing order for an element
nav-left	Specifies where to navigate when using the arrow-left navigation key
nav-right	Specifies where to navigate when using the arrow-right navigation key
nav-up	Specifies where to navigate when using the arrow-up navigation key
outline	Sets all the outline properties in one declaration
outline-color	Sets the color of an outline
outline-offset	Offsets an outline, and draws it beyond the border edge
outline-style	Sets the style of an outline
outline-width	Sets the width of an outline
resize	Specifies whether or not an element is resizable by the user
text-overflow	Specifies what should happen when text overflows the containing element

TABLE B-15 Multi-Column Layout Properties

Property	Description
break-after	Specifies the page-, column-, or region-break behavior after the generated box
break-before	Specifies the page-, column-, or region-break behavior before the generated box
break-inside	Specifies the page-, column-, or region-break behavior inside the generated box
column-count	Specifies the number of columns an element should be divided into
column-fill	Specifies how to fill columns
column-gap	Specifies the gap between the columns
column-rule	A shorthand property for setting all the column-rule-* properties
column-rule-color	Specifies the color of the rule between columns
column-rule-style	Specifies the style of the rule between columns

TABLE B-15 Multi-Column Layout Properties (Continued)

Property	Description
column-rule-width	Specifies the width of the rule between columns
column-span	Specifies how many columns an element should span across
column-width	Specifies the width of the columns
columns	A shorthand property for setting column-width and column-count
widows	Sets the minimum number of lines that must be left at the top of a page when a page break occurs inside an element

TABLE B-16 Paged Media

Property	Description
orphans	Sets the minimum number of lines that must be left at the bottom of a page when a page break occurs inside an element
page-break-after	Sets the page-breaking behavior after an element
page-break-before	Sets the page-breaking behavior before an element
page-break-inside	Sets the page-breaking behavior inside an element

TABLE B-17 Generated Content for Paged Media

Property	Description
marks	Adds crop and/or cross marks to the document
quotes	Sets the type of quotation marks for embedded quotations

TABLE B-18 Filter Effects Properties

Property	Description
filter	Defines effects (e.g. blurring or color shifting) on an element before the element is displayed

TABLE B-19 Image Values and Replaced Content

Property	Description
image-orientation	Specifies a rotation in the right or clockwise direction that a user agent applies to an image (This property is likely going to be deprecated and its functionality moved to HTML)
image-rendering	Gives a hint to the browser about what aspects of an image are most important to preserve when the image is scaled
image-resolution	Specifies the intrinsic resolution of all raster images used in/on the element
object-fit	Specifies how the contents of a replaced element should be fitted to the box established by its used height and width
object-position	Specifies the alignment of the replaced element inside its box

TABLE B-20 Masking Properties

Property	Description
mask	
mask-type	

TABLE B-21 Speech Properties

Property	Description
mark	A shorthand property for setting the mark-before and mark-after properties
mark-after	Allows named markers to be attached to the audio stream
mark-before	Allows named markers to be attached to the audio stream
phonemes	Specifies a phonetic pronunciation for the text contained by the corresponding element
rest	A shorthand property for setting the rest-before and rest-after properties
rest-after	Specifies a rest or prosodic boundary to be observed after speaking an element's content
rest-before	Specifies a rest or prosodic boundary to be observed before speaking an element's content
voice-balance	Specifies the balance between left and right channels

TABLE B-21 Speech Properties (Continued)

Property	Description
voice-duration	Specifies how long it should take to render the selected element's content
voice-pitch	Specifies the average pitch (a frequency) of the speaking voice
voice-pitch-range	Specifies variation in average pitch
voice-rate	Controls the speaking rate
voice-stress	Indicates the strength of emphasis to be applied
voice-volume	Refers to the amplitude of the waveform output by the speech synthesises

TABLE B-22 Marquee Properties

Property	Description
marquee-direction	Sets the direction of the moving content
marquee-play-count	Sets how many times the content move
marquee-speed	Sets how fast the content scrolls
marquee-style	Sets the style of the moving content

B.2 Properties Excluded From White List

Table B-23 lists the CSS Properties that are not permitted for use when building a CSS for PayPage iFrame.

TABLE B-23 CSS Properties Excluded From the White List (not allowed)

Property Name	Why excluded from white list?
background	The other properties of background like color or size can still be set with the more specific properties
background-attachment	Only makes sense in the context of background-image
background-image	Allows URL
background-origin	Only makes sense in the context of background-position
background-position	Only makes sense in the context of background-image
background-repeat	Only makes sense in the context of background-image
background-size	Only makes sense in the context of background-image
border-image	This also includes the extensions like -webkit-border-image and -o-border-image
border-image-outset	Only makes sense in the context of border-image
border-image-repeat	Only makes sense in the context of border-image
border-image-source	Allows URL
border-image-width	Only makes sense in the context of border-image
@font-face	Allows URL
list-style-image	Allows URL
cursor	Allows URL
icon	Allows URL

C

SAMPLE PAYPAGE INTEGRATION CHECKLIST

This appendix provides a sample of the PayPage Integration Checklist for use during your Implementation process. It is intended to provide information to Vantiv on your PayPage setup.

FIGURE C-1 Sample PayPage Integration Checklist

PayPage Integration Checklist

This document is intended to provide information to Vantiv on your PayPage setup. Please complete and send a copy to your Implementation Analyst prior to going live. This will be kept on file and used in the event of issues with PayPage production processing.

Merchant/Organization _____Contact Name_____

Phone_____Date Completed_____

1. **What timeout value do you plan to use in the event of a PayPage transaction timeout?**

   ____ 5000 (5 seconds) – recommended, where the timeout callback stops the transaction.

   ____ Other: _____

2. **Which unique identifier(s) do you plan to send with each PayPage Request? (Check all that apply.)**
   The values for either the <merchantTxnId> or the <orderId> must be unique so that we can use these identifiers for reconciliation or troubleshooting. You can code your systems to send either or both.

   ____ orderID

   ____ merchantTxnId

3. **What diagnostic information do you plan to collect in the event of a failed PayPage transaction? (Check all that apply.)**
   In order to assist us in determining the cause of failed PayPage transactions, we request that you collect some or all of the following diagnostic information when you encounter a failure. You will be asked to provide it to your Implementation Analyst (if you are currently in testing and certification) or Customer Support (if you are currently in production).

   ____ Error code returned and reason for the failure. For example, JavaScript was disabled on the customer's browser, or could not loaded, or did not return a response, etc.
   ____ The orderId for the transaction.

   ____ The merchantTxnId for the transaction.

   ____ Where in the process the failure occurred.

   ____ Information about the customer's browser, including the version.

Index

A

Apple Pay
  data/transaction flow, 32
  support for, 30
  using mobile API, 31
Authorizations
  request structure, 38
  response structure, 39
Availability of the PayPage API
  detecting, 20

C

Callbacks
  failure, 19
  handling, 18
  success, 18
  timeout, 20
Capture Given Auth Transactions, 48
  request structure, 48
  response structure, 50
Checkout Form Submission
  intercepting, 18
Collecting Diagnostic Information, 36
Contact Information, xii
Creating a Customized CSS, 10
Credit Transactions, 51
  request structure, 51
  response structure, 52
CSS Properties for iFrame API, 91

D

Diagnostic Information
  collecting, 36
Document Structure, x
Documentation Set, x

E

expDate, 85

F

Force Capture Transactions, 46
  request structure, 46
  response structure, 47

H

HTML Checkout Page Examples, 60
  iFrame-Integrated Checkout Page, 64
  JavaScript API-Integrated Checkout page, 61
  Non-PayPage Checkout Page, 60

I

Integration Steps, 14
Intended Audience, vii

J

JavaScript Customer Browser API, 2
jQuery Version, 8

L

LitleXML Elements, 83
Loading the PayPage API, 15

M

Migrating From Previous Versions, 6
  from JavaScript Browser API to Vantiv-Hosted iFrame, 7
  from PayPage with jQuery 1.4.2, 6
Mobile API, 2, 28
  Information Sent, 69
Mobile Application
  Integrating PayPage, 28
Mobile Operating System Compatibility, 8
Mouse Click
  handling, 17

N

Non-PayPage Checkout Page, 60

O

Online Authorization Request, 38
Online Authorization Response, 40
Online Sale Request, 41
Online Sale Response, 43
Order Handling Systems
  Information Sent, 68

P

PayPage
  Getting Started, 6
  How it works, 4
  Overview, 2
PayPage API Request Fields
  specifying, 16
PayPage API Response Fields
  specifying, 16
paypageRegistrationId, 87
POST
  Sample Response, 29
POST Request, 28
POST response, 5

R

Register Token Transactions, 44
  request structure, 44
  response structure, 45
registerTokenRequest, 88
registerTokenResponse, 89
Response Codes, 11
Revision History, vii

S

Sale Transactions, 41
  request structure, 41
  response structure, 42
Sample JavaScripts, 70
  sample PayFrame client, 75
  sample PayFrame JavaScript, 78
  sample PayPage JavaScript, 70
Sample PayPage JavaScript, 70

T

Testing and Certification, 54
Testing PayPage Transactions, 54
Transactions
  authorization, 38
  capture given auth, 48
  credit, 51
  examples, 37
  force Capture, 46
  register token, 44
  sale, 41
  types, 37
Typographical Conventions, xi

V

Vantiv-Hosted iFrame API, 2
  integrating into your checkout page, 22

X

XML Elements
  cardValidationNum, 89
  expDate, 85
  paypage, 84
  paypageRegistrationId, 87
  registerTokenRequest, 88
  registerTokenResponse, 89