Litle & Co.

PayPage Integration Guide

January 2014

XML Release: 8.24

Litle PayPage API v.2

Document Version: 2.12

Litle & Co.PayPage Integration Guide Document Version: 2.12

CONTENTS

ABOUT THIS GUIDE

This guide provides information on integrating the Litle & Co. PayPage feature, which, when used together with the Litle Vault, may eliminate your exposure to sensitive cardholder data and can significantly reduce your risk and PCI scope. It also explains how to perform PayPage transaction testing and certification with Litle & Co.

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, as well as information on PCI Compliance. 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

Document Structure

This manual contains the following sections:

Chapter 1, "Introduction"

This chapter provides an overview of the Litle & Co. 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 Litle 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 Litle PayPage feature.

Appendix B, "Sample PayPage Integration Checklist"

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

Documentation Set

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

- *Litle & Co. iQ Reporting and Analytics User Guide*
- *Litle & Co. XML Reference Guide*
- *Litle & Co. Chargeback API Reference Guide*
- *Litle & Co. Chargeback Process Guide*
- *Litle & Co. PayPal Integration Guide*
- *Litle & Co. Bill Me Later Integration Guide*
- *Litle & Co. PayFac API Reference Guide*
- *Litle & Co. PayFac Portal User Guide*
- *Litle & Co. XML Differences Guide*
- *Litle & Co. Scheduled Secure Reports Reference Guide*
- *Litle & Co. Chargeback XML and Support Documentation API Reference Guide* **(Legacy)**
- *Litle & Co. Virtual Terminal User 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 Litle & Co.

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 Litle & Co. PayPage feature. The topics discussed in this chapter are:

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

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

1.1 PayPage Overview

The goal of the Litle Vault solution is to reduce risk and PCI compliance requirements. With the Litle Vault, you no longer store the credit card number in your database. This greatly reduces your risk and PCI compliance scope, however, your order handling system still receives and processes the account number from your checkout page.

Litle PayPage extends the Vault such that your order handling system no longer handles or stores primary account numbers. The Litle PayPage feature 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 Litle PayPage JavaScript to register the provided credit card for a token. The JavaScript API validates, encrypts, and passes the account number to Litle as the first step in the form submission. When the call to Litle returns, the account number is replaced with a *PayPage Registration ID*. No card data is actually transmitted via your web server.

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

NOTE: **In order to optimally use the Litle PayPage feature as a tool for reduction in scope for PCI Requirements (i.e., to no longer handle primary account numbers), this feature must be used at all times, without exception. Failure to use this function without exception and to handle primary account numbers could potentially impact your validation type for PCI DSS Compliance and increase your scope, risk, and PCI Compliance requirements.**

FIGURE 1-1   Litle Vault

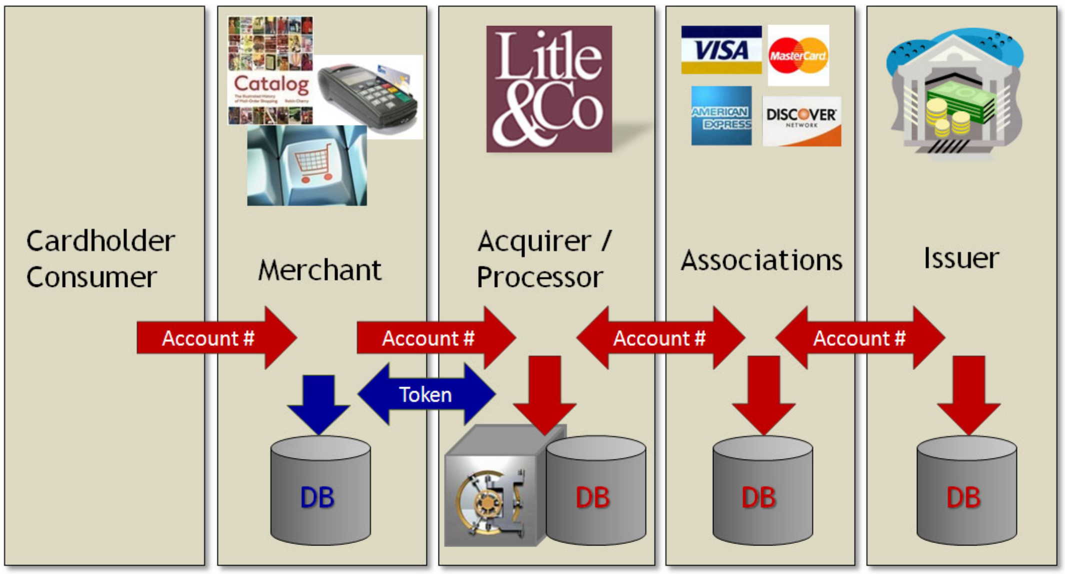
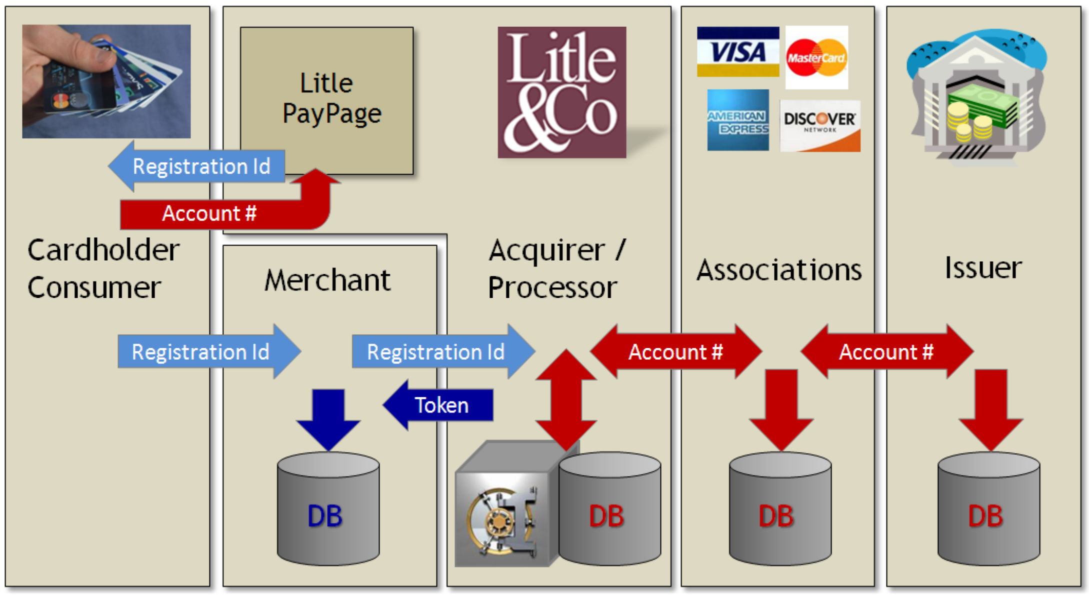
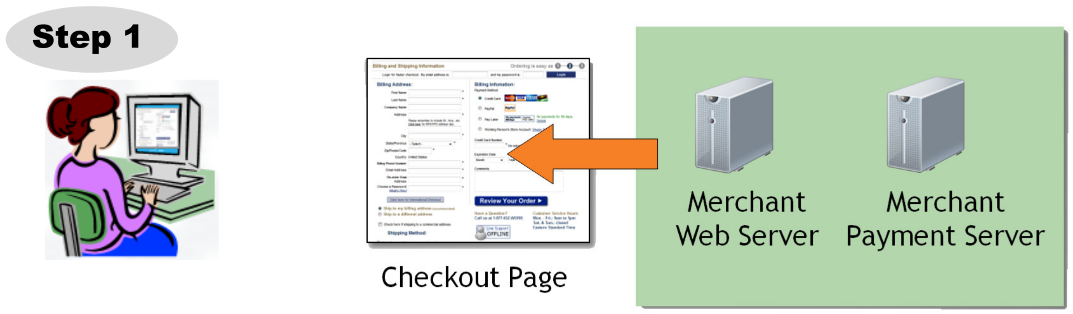


FIGURE 1-2   Litle Vault with PayPage

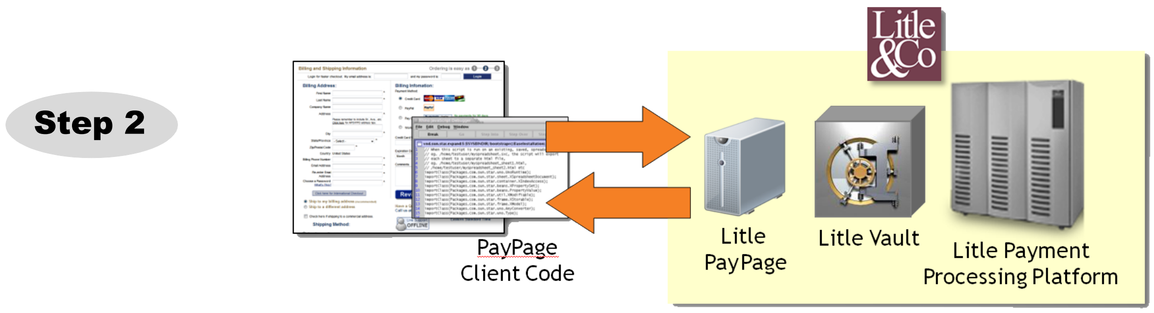


1.2 How PayPage Works

This section illustrates the Litle PayPage process.

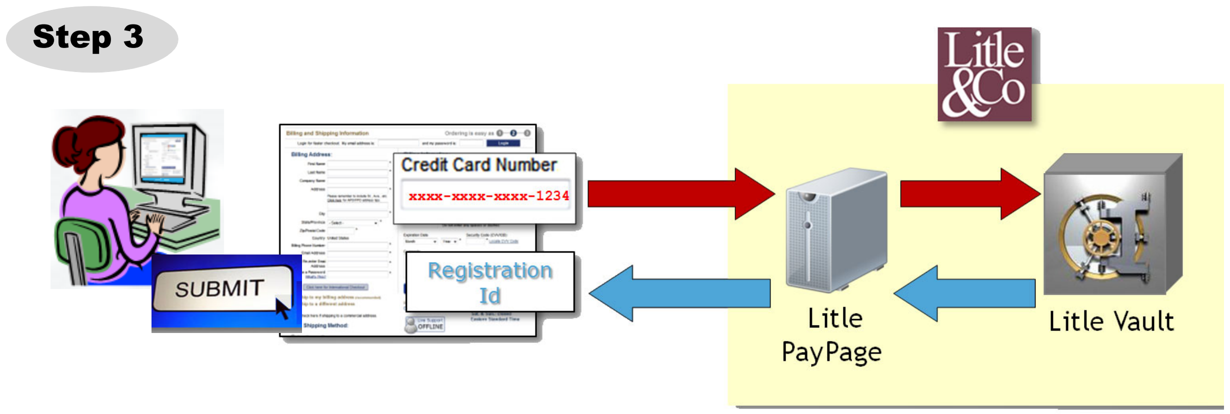


1. When your customer is ready to finalize a purchase from your website, your web server delivers your Checkout Page to the customer's web browser.

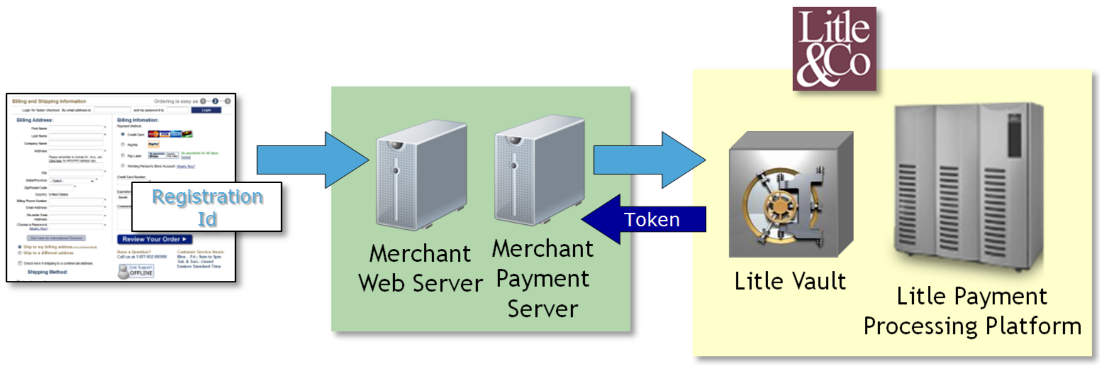


2. The customer's browser loads the PayPage Client code (JavaScript API) from the Litle PayPage server. The API validates credit cards, submits account numbers to the PayPage Service, encrypts account numbers, and adds PayPage registration IDs to the form. It also contains Litle's public key.

(continued on next page)



3. After the customer enters their card number and clicks the submit button, the PayPage API sends the card number data to the Litle PayPage service. The PayPage service submits a LitleXML transaction to the Litle Vault to register a token for the card number provided. The token is securely stored in the Litle 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.



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 to Litle with the Registration ID, and the Litle 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 Litle Vault solution, you must complete the following:

- Ensure that your organization is enabled and certified to process tokens, using the Litle Vault solution.
- Determine whether you plan to allow your order handling system to handle and store sensitive card data. See Deciding on Your Business Needs on page 12 for more information.
- Complete the Litle PayPage Integration Checklist provided by your Litle & Co. Implementation Consultant, and return to Litle Implementation. See Appendix B, "Sample PayPage Integration Checklist".
- Obtain a *PayPage ID* from Litle Implementation.
- Modify your checkout page--and any other page that receives credit card data from your customers--to integrate the PayPage feature (execute an API call to Litle). See Integrating PayPage on page 13 for more information.
- Modify your system to accept the response codes listed in Table 1-2, PayPage-Specific Response Codes Received in Browser, and Table 1-3, PayPage Response Codes Received in LitleXML Responses.
- Test and certify your checkout process. See Testing and Certification on page 45 for more information.

1.3.1 Migrating From Previous Versions of Litle PayPage API

Previous versions of the PayPage API included jQuery 1.4.2. 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 Litle 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 Litle PayPage API.
- Construct a new LitleApi module when calling `sendToLitle`.

1.3.2 Browser Compatibility

The Litle PayPage feature is compatible with the following browsers (when JavaScript is enabled):

- Mozilla Firefox 2.5 and later
- Internet Explorer 6 and later
- Safari 3 and later
- Opera 9 and later
- Chrome 1 and later

1.3.3 jQuery Version

You must load a jQuery library *before* loading the Litle API. Litle recommends using jQuery 1.4.2 or higher. Refer to http://jquery.com for more information on jQuery.

1.3.4 Litle Certification and Testing Environments

For certification and testing of Litle feature functionality, including PayPage, Litle 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 Litle merchants, and existing Litle 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 the Litle 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, Litle migrates 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	https://request-prelive.np-securepaypage-litle.com
Post-Live (Regression Testing)	JavaScript Library	https://request-postlive.np-securepaypage-litle.com/LitlePayPage/litle-api2.js
	Request Submission	https://request-postlive.np-securepaypage-litle.com
Live Production	*Production*	*Contact your Litle Implementation Consultant for the PayPage Production URL.*

1.3.5 Transitioning from Certification to Production

Before using your checkout page 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 Litle 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 PayPage-Specific Response Codes

Table 1-2 and Table 1-3 list response codes specific to the PayPage feature, received in the browser and via an LitleXML Response. For information on response codes specific to token transactions, see the *Litle & Co. XML Reference Guide*.

TABLE 1-2 PayPage-Specific Response Codes Received in Browser

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 Litle PayPage feature.

2

INTEGRATION AND TESTING

This chapter describes the steps required to integrate the Litle PayPage feature as part of your checkout page, along with information on PayPage Certification. The topics included are:

- Deciding on Your Business Needs
- Integrating PayPage
- LitleXML Transaction Examples When Using PayPage
- Testing and Certification

2.1 Deciding on Your Business Needs

When deciding how to use the Litle PayPage feature to meet your business needs, it is useful to consider the following:

- PayPage is based on JavaScript and only functions on JavaScript-enabled browsers. How will you handle a transaction if your customer disables JavaScript in their browser? Continue without the protection of PayPage or cancel the transaction?
- How will you handle a timeout that occurs during a transaction?
- In the unlikely event of a processing lapse on the Litle systems, how will you handle the transaction?

By allowing a transaction to proceed under any of the above circumstances, you are exposing your system to sensitive card data which may increase your PCI compliance risk. You must determine whether the potential loss of a sale is worth the increased risk.

In the section, "Integrating PayPage" starting on page 13, there are two sub-sections describing different coding strategies for each business model: not exposing sensitive data to your order handling systems or exposing sensitive card data to your order handling system (and falling back to a card number). Be sure to choose the appropriate model for your business needs.

2.2 Integrating PayPage

This section provides step-by-step instructions for integrating the Litle PayPage feature into your checkout page. There are two sets of instructions provided. The steps within each section are the same, however there are different coding strategies for handling certain failure scenarios, if encountered.

- **Coding Strategy 1: PayPage Implementation** – does not allow transactions to proceed when certain failure scenarios are encountered, and therefore does not expose your system to sensitive card data.
- **Coding Strategy 2: PayPage Implementation with Card Number Fallback** – allows transactions to proceed when certain failure scenarios are encountered by falling back to a card number, thereby exposing sensitive card data to your order handling system.

See Deciding on Your Business Needs on page 12 to help determine which set of instructions to follow.

NOTE: **In order to optimally use the Litle PayPage feature as a tool for reduction in scope for PCI Requirements (i.e., to no longer handle primary account numbers), “Coding Strategy 1” must be used at all times, without exception. Failure to use PayPage in this manner (handling primary account numbers) could potentially impact your validation type for PCI DSS Compliance and increase your scope, risk, and PCI Compliance requirements.**

2.2.1 Integration Steps

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

1. Loading the Litle API and jQuery
2. Specifying the Litle API Request Fields
3. Specifying the Litle 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 Litle 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 Litle 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 Litle should use to handle a successful PayPage registration.
- **errorCallback** - specifies the method Litle should use to handle a failure event (if error code is received).
- **timeoutCallback** - specifies the method Litle 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 full HTML code examples of each type of Litle PayPage implementation, see the HTML Checkout Page Examples on page 50.

2.3 Coding Strategy 1: PayPage Implementation

Implementation of the Litle PayPage feature includes the steps outlined in this section. The coding strategy in this section assumes that you do not intend to allow your order handling system to handle sensitive card data, under any circumstance. If you have chosen to allow your order handling system to be exposed to sensitive card data for certain failure scenarios, skip this section and proceed to Coding Strategy 2: PayPage Implementation with Card Number Fallback on page 22.

This section also contains information on Collecting Diagnostic Information when a failure or timeout occurs.

2.3.1 Loading the Litle API and jQuery

To load the PayPage client JavaScript library from the Litle 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 Litle 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 Litle 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. Litle recommends using version 1.4.2 or higher.

2.3.2 Specifying the Litle API Request Fields

To specify the Litle API request fields, add four hidden request fields to your checkout form for `paypageId` (a unique number assigned by Litle 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 Litle 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>
```

2.3.3 Specifying the Litle API Response Fields

To specify the Litle 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"/>
  ...
</form>
```

2.3.4 Handling the Mouse Click

In order to call the Litle 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. Litle recommends a timeout value of 5000 (5 seconds) when using this coding strategy.

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 Litle 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 Litle Implementation for the appropriate production URL.

2.3.5 Intercepting the Checkout Form Submission

Without the PayPage implementation, order data is sent to your system when the submit button is clicked. With the Litle PayPage feature, a request must be sent to the Litle 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>
```

2.3.6 Handling Callbacks for Success, Failure, and Timeout

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

2.3.6.1 Success Callbacks

The **success** callback stores the Litle response 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;
}
function submitAfterLitle (response) {
  setLitleResponseFields(response);
  document.forms['fCheckout'].submit();
 }
  ...
  </script>
 ...
</head>
```

2.3.6.2 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" on page 9). When using this coding strategy, 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 onErrorAfterLitile (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>
```

2.3.6.3 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. Litle recommends a timeout value of 5000 (5 seconds) when using this coding strategy.

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>
```

2.3.7 Detecting the Availability of the Litle 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 new LitlePayPage() != 'object') {
      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 the Litle PayPage feature is shown in Appendix A, "Code Samples and Other Information".

2.3.8 Collecting Diagnostic Information

In order to assist Litle 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 **Litle Implementation Consultant** if you are currently in the testing and certification process, or your **Litle Customer Support** 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 was loaded properly, but the `sendToLitle` call returned a response code indicating an error.
- 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 Litle Implementation Consultant if your are currently in the testing and certification process, or your Litle Relationship Manager if you are currently in production.

2.4 Coding Strategy 2: PayPage Implementation with Card Number Fallback

This section includes instructions for implementing the Litle PayPage feature with card number fallback.

The coding strategy in this section assumes that you have chosen to allow your system to handle sensitive card data for certain failure scenarios (see Deciding on Your Business Needs on page 10). If you do not plan to allow your order processing system to handle sensitive card data under any circumstance, follow the instructions in the previous section, Coding Strategy 1: PayPage Implementation on page 15.

NOTE: **In order to optimally use the Litle PayPage feature as a tool for reduction in scope for PCI Requirements (i.e., to no longer handle primary account numbers), "Coding Strategy 1" must be used at all times, without exception. Failure to use PayPage in this manner (handling primary account numbers) could potentially impact your validation type for PCI DSS Compliance and increase your scope, risk, and PCI Compliance requirements.**

2.4.1 Loading the Litle API and jQuery

To load the PayPage client JavaScript library from the Litle 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 Litle 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 Litle Implementation for the appropriate production URL.

NOTE: The example above uses a Google-hosted version of the jQuery JavaScript library. You may choose to host the library locally. Litle recommends using version 1.4.2 or higher.

2.4.2 Specifying the Litle API Request Fields

To specify the Litle API request fields, add four hidden request fields to your checkout form for `paypageId` (a unique number assigned by Litle 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 Litle 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>
```

2.4.3 Specifying the Litle API Response Fields

To specify the Litle 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 flexibility in naming 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"/>
  ...
</form>
```

2.4.4 Handling the Mouse Click

In order to call the Litle 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`. You can also trigger the `sendToLitle` request using a tab-out event.

The `sendToLitle` call includes a timeout value in milliseconds. Litle recommends a timeout value of 30000 (30 seconds) when using this coding strategy.

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 Litle 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, 30000);
            return false;
  ...
  </script>
...
</head>
```

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

2.4.5 Intercepting the Checkout Form Submission

Without the PayPage implementation, order data is sent to your system when the submit button is clicked. With the Litle PayPage feature, a request must be sent to the Litle 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. The `<noscript>` case allows you to build a page that looks and functions exactly like a non-PayPage integrated page by providing an input field tied to `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>
      <input type="submit" id="submitId" value="Check out without paypage"></input>
    </noscript>
    </td></tr>
    ...
  </table>
...
</BODY>
```

2.4.6 Handling Callbacks for Success, Failure, and Timeout

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

2.4.6.1 Success Callbacks

The **success** callback stores the Litle response 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;
}
function submitAfterLitle (response) {
  setLitleResponseFields(response);
  document.forms['fCheckout'].submit();
 }
  ...
  </script>
 ...
</head>
```

2.4.6.2 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" on page 9). When using this coding strategy, the **failure** callback sends the original account number to your order processing system (submits the order) when a non-user error is encountered.

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') {
      document.forms['fCheckout'].submit();
    }
    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') {
      document.forms['fCheckout'].submit();
  }
  return false;
  }
```

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

2.4.6.3 Timeout Callbacks

The **timeout** callback stops the transaction and sends the original account number to your order processing system (submits the order).

Timeout values are expressed in milliseconds and defined in the `sendToLitle` call, described in the section, Handling the Mouse Click on page 24. Litle recommends a timeout value of 30000 (30 seconds) when using this coding strategy.

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

2.4.7 Detecting the Availability of the Litle API

In the event that the `litle-api2.js` cannot be loaded, add the following to detect availability:

```
</BODY>
...
<script>
function callLitle() {
  if(typeof new LitlePayPage() != 'object') {
    document.forms['fCheckout'].submit();
</script>
...
</HTML>
```

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

2.5 LitleXML Transaction Examples When Using PayPage

This section describes how to format LitleXML transactions when using the PayPage feature of the Litle 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 Litle with the `<paypageRegistrationId>` returned by Litle, and the Litle 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 or Sale 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 66 for definitions of the PayPage-related elements used in these examples.

This section is meant as a supplement to the *Litle & Co. XML Reference Guide*. Refer to the *Litle & Co. XML 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 Litle PayPage feature. See the *Litle & Co. XML 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 format of the Authorization Response you receive from Litle & Co.

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="8.24" 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>Litle & Co. 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="8.24" 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 you receive from Litle & Co.

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="8.24" 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>Litle & Co. 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="8.24" 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 Litle & Co. 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`, the Litle 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 *Litle & Co. XML Reference Guide*.

Example: Online Register Token Request - PayPage

```
<litleOnlineRequest version="8.24" 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>Litle Transaction Number</litleTxnId>
  <orderId>Order Id</orderId>
  <litleToken>Litle 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 *Litle & Co. XML Reference Guide*.

Example: Online Register Token Response - PayPage

```
<litleOnlineResponse version="8.24" 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 Litle & Co. 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="8.24" 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>Litle & Co. 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="8.24" 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="8.24" 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>Litle & Co. 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="8.24" 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 the Credit request for a Litle & Co. processed transaction as follows. The structure of the request is identical for either an Online or a Batch submission.

```
<credit id="Credit Id" reportGroup="UI Report Group" customerId="Customer Id">
  <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="8.24" 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>Litle & Co. 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="8.24" 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

Litle & Co. 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 a Litle & Co. 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 JavaScript interaction as well as XML requests and responses. The PayPage Certification tests the following:

- A successful transaction
- The timeout period
- The error handler and JavaScript error codes
- LitleXML transaction requests and responses

See the section, PayPage-Specific Response Codes on page 9 for definitions of the response codes.

2.6.1 Testing PayPage Transactions

To test PayPage transactions:

1. Verify that your checkout page is coded correctly. See Integrating PayPage on page 13 for more information.
2. Verify that you are using the appropriate URL for the testing and certification environment:

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

   NOTE: This URL should only be used in the testing and certification environment. Do not use this URL in a production environment. Contact Litle Implementation for the appropriate production URL.

3. Submit transactions from your checkout page using the **Card Numbers** and **Card Validations Numbers** from Table 2-1. When performing these tests, you can use any expiration date and card type.
4. Verify that your results match the **Result** column in Table 2-1.

TABLE 2-1 Expected Browser PayPage Results

Test Case	Card Number	Card Vali-dation Number	Response Code	Result
1	5112010000000003	Any 3-digit	870 (Success)	PayPage Registration ID is generated and the card is scrubbed before the form is submitted.
2	44570100000000009	Any 3-digit	871	Checkout page displays error message to card holder, for example, “Invalid Card Number - Check and retry (not Mod10).”
3	4457010000000000000006	Any 3-digit	873	Checkout page displays error message to card holder, for example, “Invalid Card Number - Check and retry (too long).”
4	601101000003	Any 3-digit	872	Checkout page displays error message to card holder, for example, “Invalid Card Number - Check and retry (too short).”
5	44570100B00000006	Any 3-digit	874	Checkout page displays error message to card holder, for example, “Invalid Card Number - Check and retry (not a number).”
6	6011010000000003	Any 3-digit	875	Checkout page displays error message to card holder, for example, “We are experiencing technical difficulties. Please try again later or call 555-555-1212.” If you have coded your checkout page to allow your order handling system to handle credit card data, no error message is displayed and the transaction is submitted despite the error.
7	51234567898010003	Any 3-digit	876	Checkout page displays error message to card holder, for example, “Invalid Card Number - Check and retry (failure from server).”

TABLE 2-1 Expected Browser PayPage Results (Continued)

Test Case	Card Number	Card Vali-dation Number	Response Code	Result
8	375001000000005	Any 3-digit	None (Timeout error)	Checkout page displays error message to card holder, for example, “We are experiencing technical difficulties. Please try again later or call 555-555-1212 (timeout).” If you have coded your checkout page to allow your order handling system to handle credit card data, no error message is displayed and the transaction is submitted despite the error.
9	4457010200000007	Any 3-digit	889	Checkout page displays error message to card holder, for example, “We are experiencing technical difficulties. Please try again later or call 555-555-1212.” If you have coded your checkout page to allow your order handling system to handle credit card data, no error message is displayed and the transaction is submitted despite the error.
10	5112010000000003	abc	881	Checkout page displays error message to card holder, for example “Invalid Card Validation Number - Check and retry (not a number)”.
11	5112010000000003	12	882	Checkout page displays error message to card holder, for example “Invalid Card Validation Number - Check and retry (too short)”.
12	5112010000000003	12345	883	Checkout page 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 29).
2. Submit three Authorization transactions using the PayPage data from Table 2-2.
3. Verify that your `authorizationResponse` values match the **Response Code Expected** column.

TABLE 2-2 PayPage Authorization Transaction Request and Response Data

Test Case	**PayPage Registration ID**	**Exp. Date**	**Card Vali-dation 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
14	cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NWVXE4ZW5UTko4NU9KK3p1L1p1Vzg4YzVPQVlSUHNlTG1JN2I0NzlyTg==	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 Litle PayPage Solution. The following sections are included:

- HTML Checkout Page Examples
- Information Sent to Order Processing Systems
- Sample Litle JavaScript (litle-api2.js)
- LitleXML Elements for PayPage

A.1 HTML Checkout Page Examples

This section provides three HTML checkout page examples:

- HTML Example for Non-PayPage Checkout Page
- HTML Example for PayPage-Integrated Checkout Page
- HTML Example for Checkout Page with Card Number Fallback Implementation

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 the Litle PayPage feature. 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>&nbsp;</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 PayPage-Integrated Checkout Page

The HTML code below is an example of a simple checkout page integrated with the Litle PayPage feature. This example assumes that you do not intend to allow your order handling system to handle sensitive card data (see Coding Strategy 1: PayPage Implementation on page 15).

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 Litle 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;
        }
        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
       or call 555-555-1212");
              }
```

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

```
                  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"
size="20"></td></tr>
      <tr><td>Last Name</td><td><input type="text" id="lName" name="lName"
```

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

```
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>&nbsp;</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"/>
  </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 Checkout Page with Card Number Fallback Implementation

The HTML code below is an example of a simple checkout page integrated with the Litle PayPage feature. This example assumes that you have chosen to allow your order handling system to handle sensitive card data thereby exposing your systems to sensitive card data (and falling back to a card number). See Coding Strategy 2: PayPage Implementation with Card Number Fallback on page 22 for more information.

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 Litle 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;
        }
        function submitAfterLitle (response) {
          setLitleResponseFields(response);
          document.forms['fCheckout'].submit();
        }
        function timeoutOnLitle () {
          document.forms['fCheckout'].submit();
        }

            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') {
```

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

```
                  document.forms['fCheckout'].submit();
                }
                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') {
                document.forms['fCheckout'].submit();
            }
            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, 30000);
                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"
size="20"></td></tr>
```

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

```
      <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="cvv2" name="cvv2" size="5"></td></tr>
      <tr><td>Exp Date</td><td><input type="text" id="expDate" name="expDate" 
size="5"></td></tr>
      <tr><td>&nbsp;</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>
          <input type="submit" id="submitId" value="Check out without PayPage"></input>
      </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"/>
  </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') {
      document.forms['fCheckout'].submit();
  }

}
</script>
</HTML>
```

A.2 Information Sent to Order Processing Systems

Depending upon which coding strategy you used to create your checkout page (see Integrating PayPage on page 13), different information is sent to your order handling system after failure and timeout scenarios.

A.2.1 Information Sent Without Integrating Litle PayPage

If you have already integrated the Litle 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 - 4100000000000000001
lName - Buyer
```

A.2.2 Information Sent with PayPage Integration

When you code your checkout page to stop a transaction when a failure or timeout occurs (see Coding Strategy 1: PayPage Implementation on page 15), you are not exposing your order processing system to sensitive card data. The success callback stores the Litle 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

A.2.3 Information Sent with Card Number Fallback Implementation

When you code your checkout page to continue with a transaction when a failure or timeout occurs (see Coding Strategy 2: PayPage Implementation with Card Number Fallback on page 22), you expose your order processing system to sensitive card data. The success callback stores the Litle response 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.

The **failure** callback, however, posts the original account number. This is an example of information sent to your order handling system:

cvv - 123
expDate - 1210
fName - Joe
ccNum - 4100000000000001
lName - Buyer
request$paypageId - a2y4o6m8k0
request$merchantTxnId - 987012
request$orderId - order_123
request$reportGroup - *merchant1500
response$paypageRegistrationId
response$bin
response$code - 889
response$message - Failure
response$responseTime - 2010-12-21T12:45:15Z
response$type
response$litleTxnId - 21200000051806

Similarly, the **timeout** callback posts the original account number. This is an example of information sent to your order handling system:

cvv - 123
expDate - 1210
fName - Joe
ccNum - 4100000000000001
lName - Buyer
request$paypageId - a2y4o6m8k0
request$merchantTxnId - 987012
request$orderId - order_123
request$reportGroup - *merchant1500
response$paypageRegistrationId
response$bin
response$code

```
response$message
response$responseTime
response$type
response$litleTxnId
```

A.3 Sample Litle JavaScript (litle-api2.js)

The following is an example of a complete Litle JavaScript with a sample public key.

```
/*!
 * Litle & Co. Pay Page Java Script API Version: 2.0
 * Copyright © 2003-2013, 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);
                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.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 Litle & Co. transaction processing systems. Each section defines a particular element, its relationship to other elements (parents and children), as well as any attributes associated with the element.

For additional information on the structure of LitleXML requests and responses using these elements, as well as XML examples, see LitleXML Transaction Examples When Using PayPage on page 28. For a comprehensive list of all LitleXML elements and usage, see Chapter 4, “LitleXML Elements” in the *Litle & Co. XML 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`, the Litle 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.

Type = String; **minLength** = N/A; **maxLength** = 4

Parent Elements:

card, paypage, token, registerTokenRequest, updateCardValidationNumOnToken

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.

Litle & Co. recommends 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 Litle Vault solution.

Parent Elements:

authorization, sale, captureGivenAuth, forceCapture, credit

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 (Litle 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 the user interface where this transaction displays. **minLength** = 1 **maxLength** = 25

Child Elements:

Required: (choice of) accountNumber, echeckForToken, 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 Litle 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

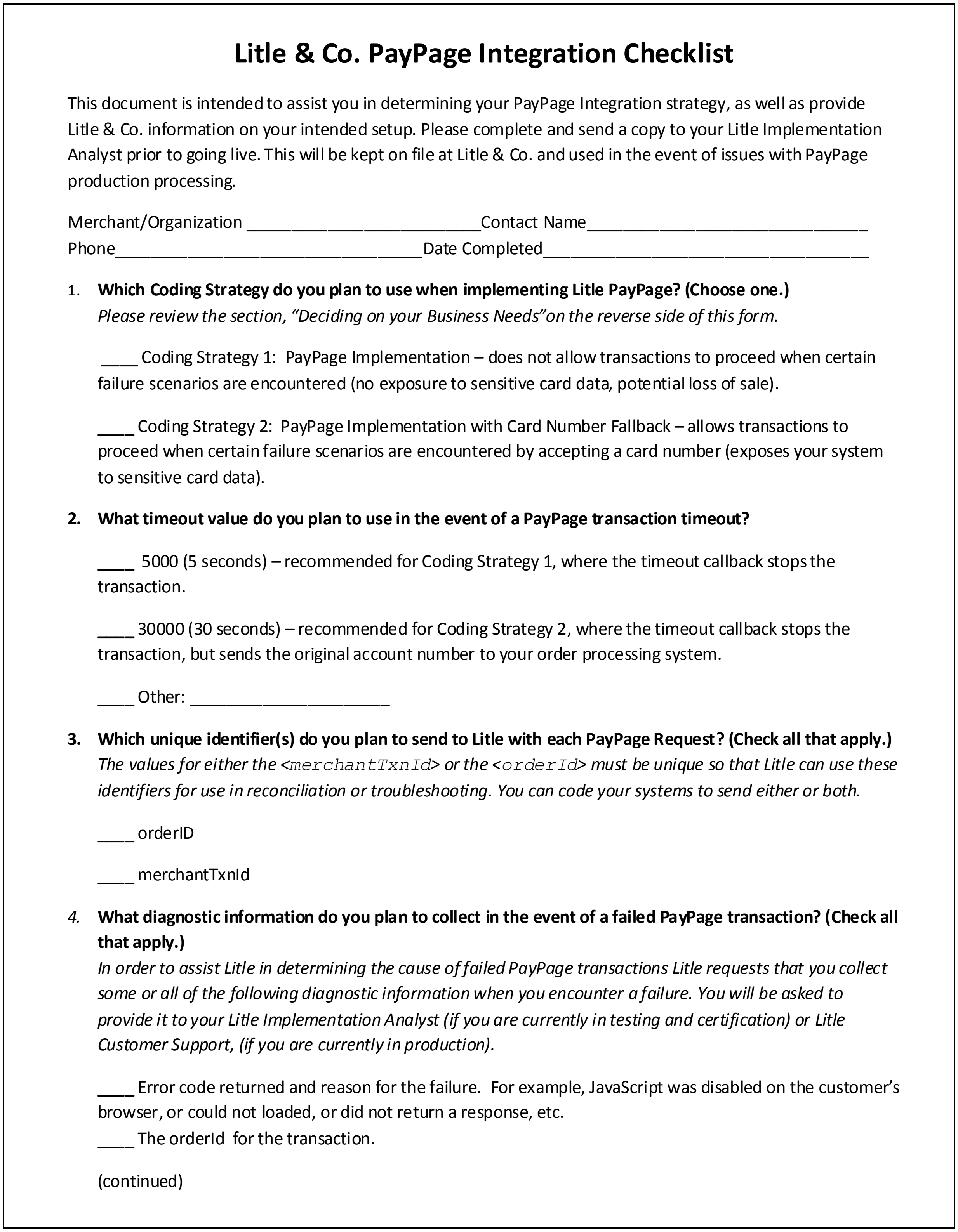
Optional: eCheckAccountSuffix, orderId, litleToken, bin, type

B

SAMPLE PAYPAGE INTEGRATION CHECKLIST

This appendix provides a sample of the PayPage Integration Checklist for use during your Implementation process. It is intended to assist you in determining your PayPage Integration strategy, as well as provide Litle & Co. information on your intended setup.

FIGURE B-1 Sample PayPage Integration Checklist



Litle & Co. PayPage Integration Checklist

This document is intended to assist you in determining your PayPage Integration strategy, as well as provide Litle & Co. information on your intended setup. Please complete and send a copy to your Litle Implementation Analyst prior to going live. This will be kept on file at Litle & Co. and used in the event of issues with PayPage production processing.

Merchant/Organization ___________________________Contact Name________________________________
Phone___________________________________Date Completed_____________________________________

1. **Which Coding Strategy do you plan to use when implementing Litle PayPage? (Choose one.)**
Please review the section, "Deciding on your Business Needs"on the reverse side of this form.

____ Coding Strategy 1: PayPage Implementation – does not allow transactions to proceed when certain failure scenarios are encountered (no exposure to sensitive card data, potential loss of sale).

____ Coding Strategy 2: PayPage Implementation with Card Number Fallback – allows transactions to proceed when certain failure scenarios are encountered by accepting a card number (exposes your system to sensitive card data).

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

____ 5000 (5 seconds) – recommended for Coding Strategy 1, where the timeout callback stops the transaction.

____ 30000 (30 seconds) – recommended for Coding Strategy 2, where the timeout callback stops the transaction, but sends the original account number to your order processing system.

____ Other: ______________________

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

____ orderID

____ merchantTxnId

4. **What diagnostic information do you plan to collect in the event of a failed PayPage transaction? (Check all that apply.)**
In order to assist Litle in determining the cause of failed PayPage transactions Litle requests 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 Litle Implementation Analyst (if you are currently in testing and certification) or Litle 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.

(continued)

____ The merchantTxnId for the transaction.

____ Where in the process the failure occurred.

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

Deciding on Your Business Needs

When deciding how to use the Litle PayPage feature to meet your business needs, it is useful to consider the following:

- PayPage is based on JavaScript and only functions on JavaScript-enabled browsers. How will you handle a transaction if your customer disables JavaScript in their browser? Continue without the protection of PayPage or cancel the transaction?
- How will you handle a timeout that occurs during a transaction?
- In the unlikely event of a processing lapse on the Litle systems, how will you handle the transaction?

By allowing a transaction to proceed under any of the above circumstances, you are exposing your system to sensitive card data which may increase your PCI compliance risk. You must determine whether the potential loss of a sale is worth the increased risk.

Coding Strategies

In the section, "Integrating PayPage" of the *Litle & Co. PayPage Integration Guide*, there are two sub-sections describing different coding strategies for each business model. Be sure to choose the appropriate model for your business needs:

- **Coding Strategy 1:** PayPage Implementation – does not allow transactions to proceed when certain failure scenarios are encountered, and therefore does not expose your system to sensitive card data.
- **Coding Strategy 2:** PayPage Implementation with Card Number Fallback – allows transactions to proceed when certain failure scenarios are encountered by falling back to a card number, thereby exposing sensitive card data to your order handling system.

Note: In order to optimally use the Litle PayPage feature as a tool for reduction in scope for PCI Requirements (i.e., to no longer handle primary account numbers), "Coding Strategy 1" must be used at all times, without exception. Failure to use PayPage in this manner (handling primary account numbers) could potentially impact your validation type for PCI DSS Compliance and increase your scope, risk, and PCI Compliance requirements.

Index