eProtect™ Integration Guide

January 2018

cnpAPI Release: V12.0

eProtect API V2.1

Document Version: 6.6

Vantiv eProtect™ Integration Guide Document Version: 6.6

CONTENTS

About This Guide

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

NOTE: **The PayPage product is now known as *Vantiv eProtect*. The term 'PayPage' however, is still used in this guide in certain text descriptions, along with many data elements, JS code, and URLs. Use of these data elements, etc., with the PayPage name is still valid with this release, but will transition to 'eProtect' in a future release.**

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

TABLE 1 Document Revision History (Continued)

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

TABLE 1 Document Revision History (Continued)

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

TABLE 1 Document Revision History (Continued)

Doc. Version	Description	Location(s)
4.4	Corrected the URL for in various examples for mobile POST requests from: https://request-prelive.np-securepaypage-litle.com/LitlePayPage to https://request-prelive.np-securepaypage-litle.com/LitlePayPage **/paypage**	Chapter 1 and 2
4.5	Updated the guide to include information on the Vantiv-Hosted PayPage iFrame solution (many changes and re-arrangement of sections). Also includes the addition of Appendix B, and new HTML and JavaScript samples in Appendix A.	All
4.6	Updated callback handling code examples for iFrame.	Chapter 2
4.7	Added a link for accessing an example iFrame page.	Chapter 1
	Added a new parameter for iFrame applications - 'htmlTimeout,' as well as two new error codes for failures when the iFrame fails to load (884), or if the CSS fails load (885). Updated the code examples to reflect these additions.	Chapter 1 and Chapter 2
	Updated information on testing URLs and User Agent examples for Mobile applications.	Chapter 2
4.8	Updated the Browser and Mobile Operating System Compatibility section (removed support for Android 2.1 and 2.2).	Chapter 1
	Added a step (Step 7) on creating a style sheet to the section on migrating from JavaScript Browser API to Vantiv-Hosted iFrame.	Chapter 1
	Updated the Apple Pay™ POST Response to include an expiration date.	Chapter 2
4.9	Corrected the iFrame URLs in Table 1-1 from: https://request-prelive.np-securepaypage-litle.com/LitlePayPage /payframe-client.min.js to https://request-prelive.np-securepaypage-litle.com/LitlePayPage **/js**/payframe-client.min.js.	Chapter 1
5.0	Updated product name in applicable areas throughout the guide from *PayPage* to *eProtect* (Phase 1). Also updated many instances of *PayFrame* to *iFrame*.	All
	Updated the eProtect workflow diagrams and steps.	Chapter 1
	Updated LitleXML examples to reflect updates to LitleXML V10.0.	Chapter 2

TABLE 1 Document Revision History (Continued)

Doc. Version	Description	Location(s)
5.1	Added information on obtaining CSS sample files from Vantiv.	Chapter 1 and Appendix B
	Increased recommended transaction timeout value to 15000 (15 seconds) from 5000 (five seconds).	Chapter 2, Appendix A and C
5.2	Added information on maximum length and data type for `orderId` and `id` parameters.	Chapter 2
	Added information on the length of time the CVV values are held (24 hours).	Chapter 2
	Removed all references to eChecks (not currently supported for eProtect).	Chapter 2
	Updated/corrected information in the testing sections.	Chapter 2
5.3	Removed the sample JavaScripts in Appendix A. Sample scripts are available at the pre-live, post-live, and production eProtect URLs.	Chapter 1, Appendix A
5.4	Added information on support for Apple Pay on the Web.	Chapters 1, 2, Appendix A
	Added code and definitions for `litleFormFields` to the Browser JavaScript API example.	Chapter 2
5.5	Corrected the reference to ‘Apple PassKit Framework’ to ‘Apple Pay JavaScript API’ in Step 1 of the Apple Pay Web process.	Chapter 2
	Updated the Apple PKPayment Token documentation URL.	Chapter 2
5.6	Added notes to LitleXML transaction examples related to expiration dates (required for eProtect transactions).	Chapter 2
5.7	Updated contact e-mails, phone numbers, and hours of operation in the Contact Information section.	Preface
	Added information on loading the JavaScript API (must be loaded daily to your checkout page).	Chapter 2
5.8	Changed some text instances of ‘PayFrame’ to ‘iFrame.’	Chapter 2
	Added notes to recommend eProtect testing using different devices and all browsers.	Chapter 1 and 2

TABLE 1 Document Revision History (Continued)

Doc. Version	Description	Location(s)
5.9	Added information on support for Android Pay.	Chapter 1 and 2
	Updated the XML examples to reference LitleXML 11.0.	Chapter 2
	Updated the descriptions of `id` and `orderId` fields/elements.	Chapter 2
	Added recommendation on avoiding ‘Flash of Un-styled Content’ (FOUC) issues.	Chapter 2
6.0	Updated all URLs (JavaScript library request, submission request, etc.) due to retirement of Litle domain.	Chapter 1 and 2
	Added note on informing customer of JavaScript requirement.	Chapter 2
	Updated most instances of ‘PayFrame’ with ‘iFrame.’	All
	Updated numerous function and object names (due to retirement of Litle name) with Vaniv, Vantivcnp or eProtect, etc.	All
	Updated cnpAPI version to 11.1.	Chapter 2
6.1	Added information on new enhancements to iFrame for greater iFrame and CSS customizations, including in-line field validations, tool tip additions and customizations, etc.	Chapter 1 and Chapter 2
6.2	Updated the *testlitle.com* sample page URLs to *testvantivcnp.com.* Added further information on which sample page URL to view when using the new enhanced iFrame features.	Chapter 1
6.3	Corrected the Request Submission URL in Table 1-2 from *https://request-eprotect.vantivprelive.com* to *https://request.eprotect.vantivprelive.com*.	Chapter 1
	Added information on Pre-Live Certification Environment maintenance and limitations.	Chapter 1
	Updated cnpAPI version to 11.2.	Chapter 2
6.4	Added information on required communication protocol.	Chapter 1
	Corrected some instance of ‘eProtect’ in code samples.	Appendix A

TABLE 1 Document Revision History (Continued)

Doc. Version	Description	Location(s)
6.5	Updated all cnpAPI element names to replace *Litle* with *cnp*. For example, *litleToken* is now *cnpToken*, *litleOnlineRequest* is now *cnpOnlineRequest*, etc. This includes the namespace: *http://www.litle.com/schema* is now *http://www.vantivcnp.com/schema.*	Chapter 2
	Corrected various eProtect element spelling errors, cleaned up miscellaneous coding in the HTML examples.	Chapter 2 and Appendix A
	Re-worked Section 1.3.8, Creating a Customized CSS for iFrame for clarification.	Chapter 1
	Added information on new CSS-allowed Appearance properties.	Appendix B
6.6	Re-arranged and relocated the *Creating a Customized CCS for iFrame* section for better understanding.	Chapter 1
	Added information on including a 'Trust Icon' on your payment page when customizing iFrame CSS files.	Chapter 1
	Removed workaround information for Flash of Unstyled Content (FOUC) as this issue has been corrected.	Chapter 2
	Added information on Pay With Google.	Chapter 1 and 2

Document Structure

This manual contains the following sections:

Chapter 1, "Introduction"

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

Chapter 2, "Integration and Testing"

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

Appendix A, "Code Samples and Other Information"

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

Appendix B, "CSS Properties for iFrame API"

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

Appendix C, "Sample eProtect Integration Checklist"

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

Documentation Set

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

- *Vantiv eCommerce iQ Reporting and Analytics User Guide*
- *Vantiv cnpAPI Reference Guide*
- *Vantiv eCommerce Solution for Apple Pay*
- *Vantiv Chargeback API Reference Guide*
- *Vantiv Chargeback Process Guide*
- *Vantiv PayPal Integration Guide*
- *Vantiv PayFac API Reference Guide*
- *Vantiv PayFac Portal User Guide*
- *Vantiv cnpAPI Differences Guide*
- *Vantiv Scheduled Secure Reports Reference Guide*
- *Vantiv Chargeback XML and Support Documentation API Reference Guide* **(Legacy)**

Typographical Conventions

Table 2 describes the conventions used in this guide.

TABLE 2 Typographical Conventions

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.

TABLE 2 Typographical Conventions

Convention	Meaning
. . .	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) • cnpAPI elements
[ ]	Brackets enclose optional clauses from which you can choose one or more option.
bold text	Bold text indicates emphasis.
Italicized text	Italic type in text indicates a term defined in the text, the glossary, or in both locations.
blue text	Blue text indicates a hypertext link.

Contact Information

This section provides contact information for organizations within Vantiv

Implementation - For certification and technical issues concerning your implementation of cnpAPI or issues encountered during the on-boarding process, call your assigned Implementation Consultant or e-mail to the address below.

Implementation Contact Information

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

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

Relationship Management/Customer Service Contact Information

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

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

Chargebacks Department Contact Information

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

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

Technical Publications Contact Information

E-mail	TechPubs@vantiv.com

1

INTRODUCTION

NOTE: **The PayPage product is now known as *Vantiv eProtect*™. The term 'PayPage' however, is still used in this guide in certain text descriptions, along with many data elements, JS code, and URLs. Use of these data elements, etc., with the PayPage name is still valid with this release, but will transition to 'eProtect' in a future release.**

This chapter provides an introduction and an overview of Vantiv eProtect™. The topics discussed in this chapter are:

- eProtect Overview
- How eProtect Works
- Getting Started with eProtect
  - Migrating From Previous Versions of the eProtect API
  - eProtect Support for Apple Pay™ / Apple Pay on the Web
  - eProtect Support for Android Pay™
- Creating a Customized CSS for iFrame

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

1.1 eProtect Overview

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

Vantiv provides three integration options for eProtect:

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

For more information on PCI compliance and the Vantiv eProtect product, see the *Vantiv eProtect iFrame Technical Assessment Paper*, prepared by Coalfire IT Audit and Compliance.

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

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

FIGURE 1-1 Vault

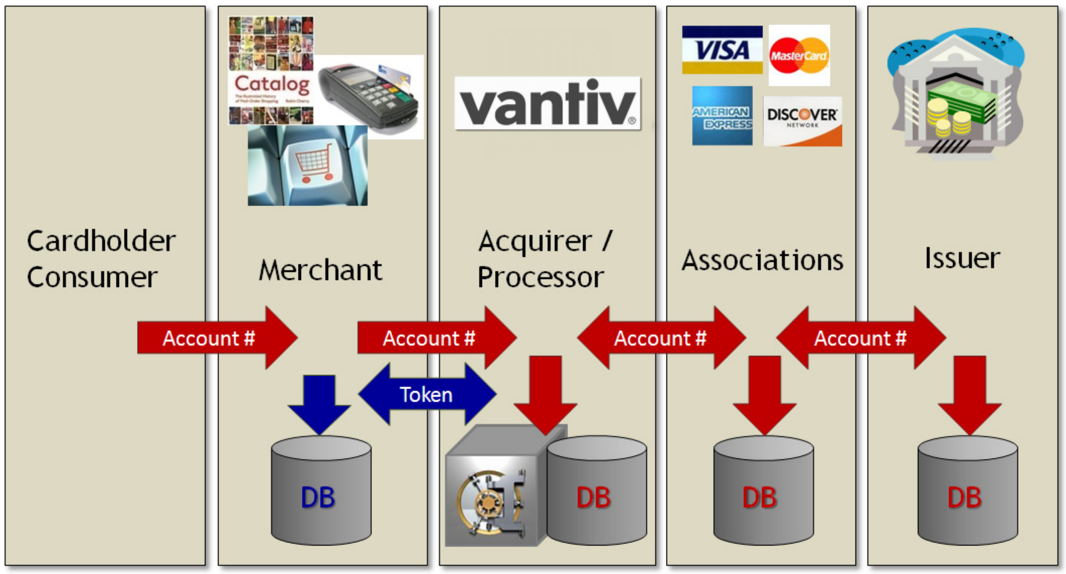
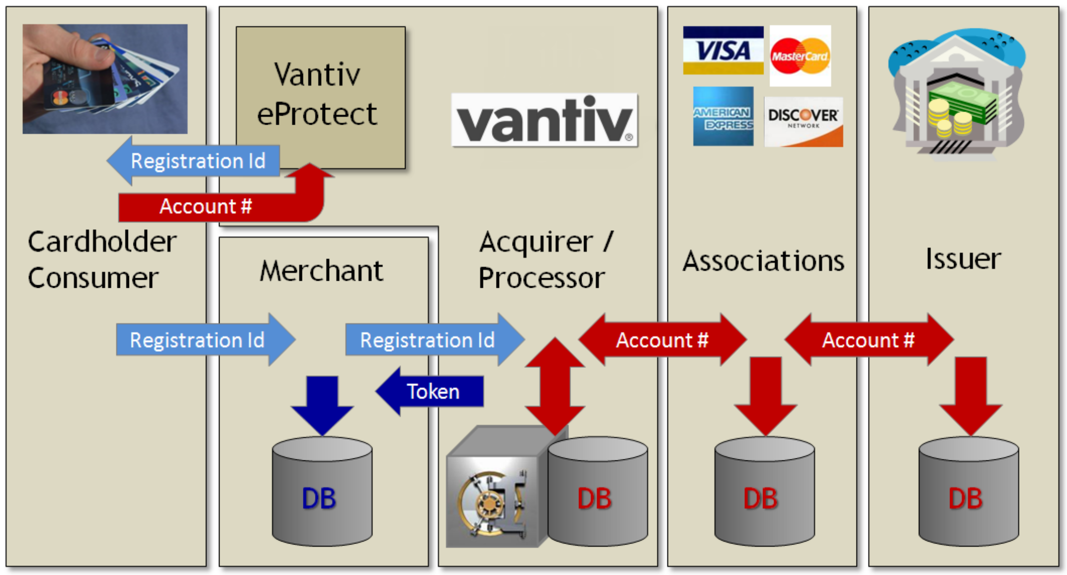
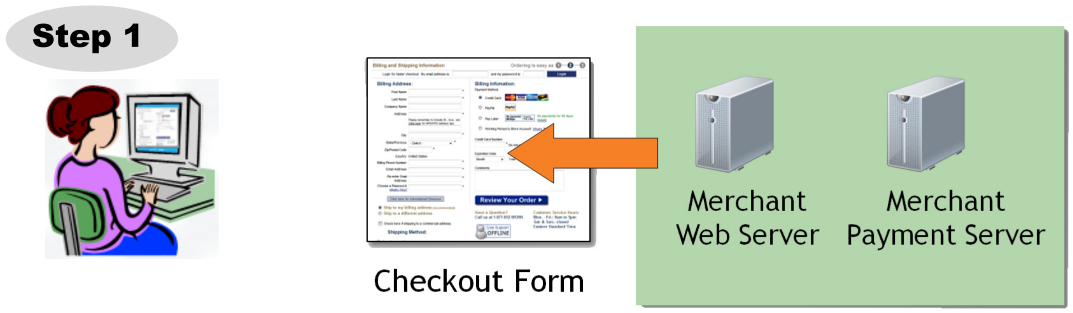


FIGURE 1-2 eProtect

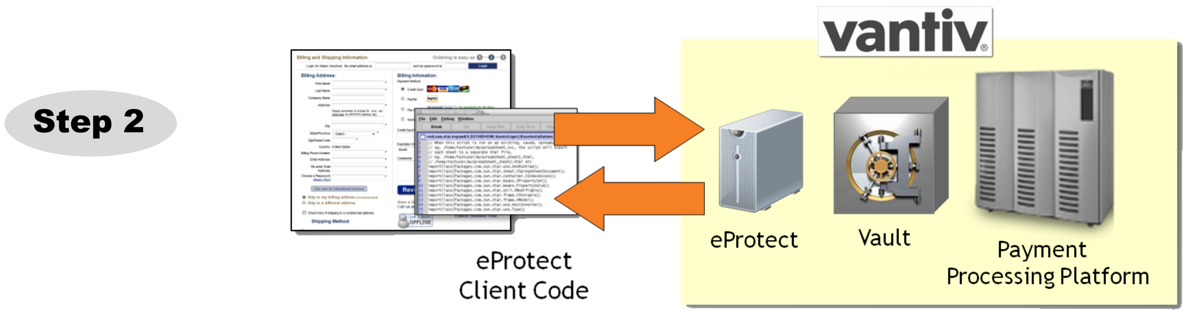


1.2 How eProtect Works

This section illustrates the eProtect process.

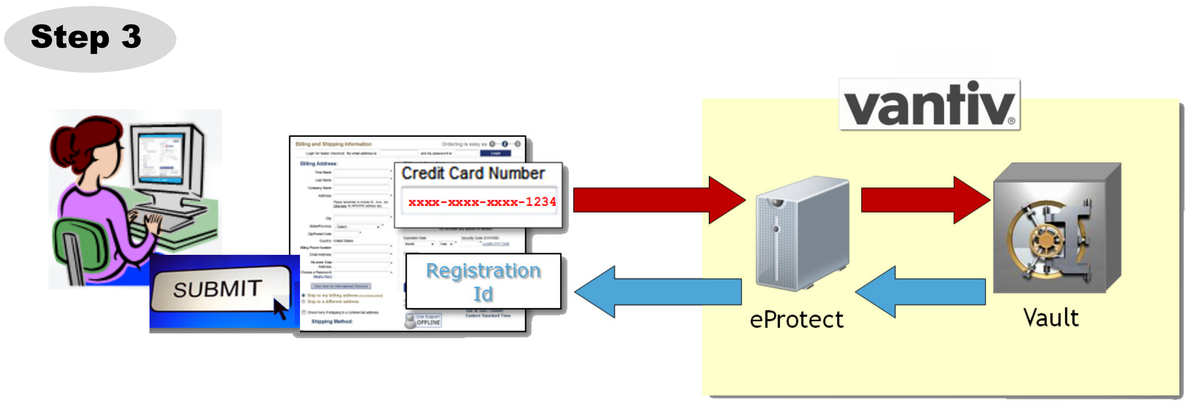


1. When your customer is ready to finalize a purchase from your website or mobile application, your web server delivers your Checkout Form to the customer's web browser or mobile device.

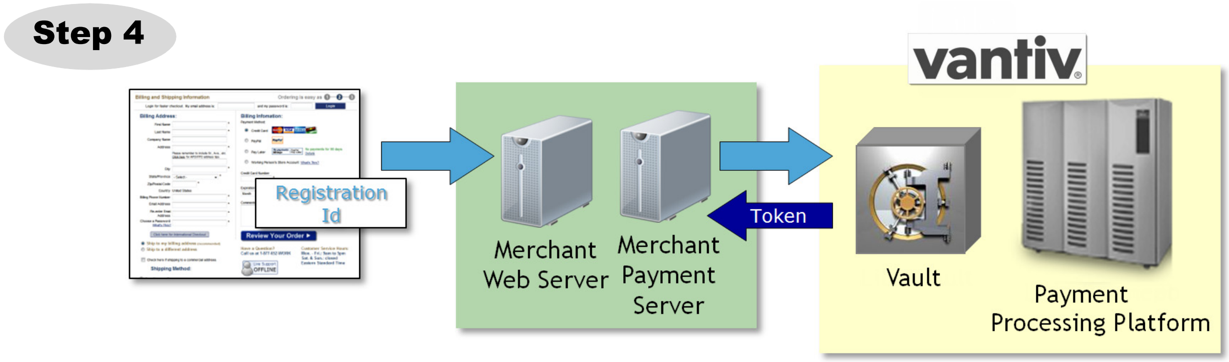


2. If using the iFrame API, the browser loads the iFrame URL hosted by the eProtect server. If using the JavaScript API, the browser loads the eProtect Client code (JavaScript) from the eProtect server. The API validates credit cards, submits account numbers to the eProtect Service, encrypts account numbers, and adds the Registration IDs to the form. It also contains Vantiv' s public encryption key.

(continued on next page)



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



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

1.3 Getting Started with eProtect

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

- Ensure that your organization is enabled and certified to process tokens, using the Vault solution.
- Complete the eProtect Integration Checklist provided by your Implementation Consultant, and return to Implementation. See Appendix C, "Sample eProtect Integration Checklist".
- Obtain a *PayPage ID* from our Implementation department.
- If you are implementing the iFrame solution, create a Cascading Style Sheet (CSS) to customize the look and feel of the iFrame to match your checkout page, then submit the Style Sheet to Vantiv for verification. See Creating a Customized CSS for iFrame on page 14 for more information.
- Modify your checkout page or mobile native application--and any other page that receives credit card data from your customers--to integrate the eProtect feature (execute an API call or POST to our system). See one of the following sections, depending on your application:
  - Integrating Customer Browser JavaScript API Into Your Checkout Page on page 22
  - Integrating iFrame into your Checkout Page on page 32
  - Integrating eProtect Into Your Mobile Application on page 39 for more information.
- Modify your system to accept the response codes listed in Table 1-3, eProtect-Specific Response Codes Received in Browsers or Mobile Devices, and Table 1-4, eProtect Response Codes Received in cnpAPI Responses.
- Test and certify your checkout process. See Testing and Certification on page 69 for more information.

1.3.1 Migrating From Previous Versions of the eProtect API

1.3.1.1 From eProtect with jQuery 1.4.2

Previous versions of the eProtect API included jQuery 1.4.2 (browser-based use only). Depending on the implementation of your checkout page and your use of other versions of jQuery, this may result in unexpected behavior. This document describes version 2 of the eProtect 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 eProtect API.
- Construct a new eProtect API module when calling `sendToEprotect`.

1.3.1.2 From JavaScript Browser API to iFrame

When migrating from the JavaScript Customer Browser API eProtect solution to the iFrame solution, complete the following steps. For a full HTML code example a iFrame eProtect implementation, see the HTML Example for Hosted iFrame-Integrated Checkout Page on page 78.

1. Remove the script that was downloading `eProtect-api2.js`.
2. Add a script tag to download `eprotect-iframe-client.min.js`.
3. On your form, remove the inputs for account number, cvv, and expiration date. Put an empty `div` in its place.
4. Consolidate the three callbacks (`submitAfterEprotect`, `onErrorAfterEprotect` and `onTimeoutAfterEprotect` in our examples) into one callback that accepts a single argument. In our example, this is called `eProtectiframeClientCallback`.
5. To determine success or failure, inspect `response.response` in your callback. If successful, the response is '870.' Check for time-outs by inspecting the `response.timeout`; if it is defined, a timeout has occurred.
6. In your callback, add code to retrieve the `paypageRegistrationId`, `bin`, `expMonth` and `expYear`. Previously, `paypageRegistrationId` and `bin` were placed directly into your form by our API, and we did not handle `expMonth` and `expYear` (we've included these inside our form to make styling and layout simpler).
7. Create a Cascading Style Sheet (CSS) to customize the look and feel of the iFrame to match your checkout page, then submit the Style Sheet to Vantiv for verification. See Creating a Customized CSS for iFrame on page 14 and Configuring the iFrame on page 33 for more information.
8. See Calling the iFrame for the Registration ID on page 35 to retrieve the `paypageRegistrationId`.

1.3.2 Browser and Mobile Operating System Compatibility

The eProtect feature is compatible with the following (see Table 1-1, "Apple Pay on the Web Compatible Devices" for information on Apple Pay web):

Browsers (when JavaScript is enabled):

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

Native Applications on **Mobile Operating Systems:**

- Chrome Android 40 and later
- Android 2.3 and later
- Apple iOS 3.2 and later
- Windows Phone 10 and later
- Blackberry 7, 10 and later
- Other mobile OS

IMPORTANT: **Because browsers differ in their handling of eProtect transactions, Vantiv recommends testing eProtect on various devices (including smart phones and tablets) and all browsers, including Internet Explorer/Edge, Google Chrome, Apple Safari, and Mozilla Firefox.**

1.3.2.1 Communication Protocol Requirement

If you are using an MPLS network, Vantiv requires that you use TLS 1.2 encryption.

1.3.3 eProtect Support for Apple Pay™ / Apple Pay on the Web

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

If you wish to allow Apple Pay transactions from your native iOS mobile applications, you must build the capability to make secure purchases using Apple Pay into your mobile application. The operation of Apple Pay on the iPhone and iPad is relatively simple: either the development of a new native iOS application or modification of your existing application that includes the use of the Apple PassKit Framework, and the handling of the encrypted data returned to your application by Apple Pay. See Using the Vantiv Mobile API for Apple Pay on page 42 for more information.

For **Apple Pay on the Web**, integration requires that the `<applepay>` field be included in the `sendToEprotect` call when constructing your checkout page with the JavaScript Customer Browser API. See Integrating Customer Browser JavaScript API Into Your Checkout Page on page 22 and Using the Customer Browser JavaScript API for Apple Pay on the Web on page 30 for more information on the complete process. Also, see Table 1-1, Apple Pay on the Web Compatible Devices for further information on supported Apple devices.

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

TABLE 1-1 Apple Pay on the Web Compatible Devices

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

1.3.4 eProtect Support for Android Pay™

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

Vantiv supports two methods for merchants to submit Android Pay transactions from Web/Mobile applications to the eCommerce platform. The preferred method involves you sending certain Vantiv specific parameters to Google®. The response from Google includes a Vantiv `paypageRegistrationId`, which you use in you payment transaction submission to Vantiv. With the alternate method, you receive encrypted information from Google, decrypt it on your servers, and submit the information to Vantiv in a payment transaction. See Using the Vantiv Mobile API for Android Pay on page 45 for more information.

1.3.5 eProtect Support for Pay with Google™

Pay With Google is an on-line payment method that lets your customers use the cards they've saved to their Google Account to pay quickly and easily in your apps. and on your websites. By clicking the Pay with Google button, customers can choose a payment method saved in their Google Account and finish checkout in a few, simple steps.

You can use the Google Payment API to simplify payments for customers who make purchases in Android apps or on Chrome with a mobile device.

Vantiv supports two methods for merchants to submit Pay with Google transactions from Mobile applications to the eCommerce platform. The preferred method involves you sending certain Vantiv-specific parameters to Google. The response from Google includes a Vantiv `paypageRegistrationId`, which you use normally in your payment transaction submission to Vantiv. With the alternate method, you receive encrypted information from Google, decrypt it on your servers, and submit the information to Vantiv in a payment transaction. See Using the Vantiv Mobile API for Pay with Google on page 48 for more information.

1.3.6 jQuery Version

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

1.3.7 Certification and Testing Environments

For certification and testing of Vantiv feature functionality, including eProtect, Vantiv uses two certification and testing environments:

- **Pre-Live** - this test environment is used for all merchant Certification testing. This environment should be used by both newly on-boarding Vantiv merchants, and existing merchants seeking to incorporate additional features or functionalities (for example, eProtect) into their current integrations.
- **Post-Live** - this test environment is intended for merchants that are already fully certified and submitting transactions to our Production platform, but wish to perform regression or other tests to confirm the operational readiness of modifications to their own systems. Upon

completion of the initial certification and on-boarding process, we migrate merchants that are Production-enabled to the Post-Live environment for any on-going testing needs.

Use the URLs listed in Table 1-2 when testing and submitting eProtect transactions. **Sample JavaScripts** are available at pre-live, post-live, and production eProtect URLs. The following sample scripts are available:

- eProtect JavaScript (eProtect-api2.js)
- iFrame Client (eprotect-iframe-client.js)
- iFrame JavaScript (eProtect-iframe.js)

TABLE 1-2 eProtect Certification, Testing, and Production URLs

Environme nt	URL Purpose	URL
Pre-Live (Testing and Certification)	JavaScript Library	https://request.eprotect.vantivprelive.com/eProtect/eProtect-api2.js
	Request Submission (excluding POST)	https://request.eprotect.vantivprelive.com
	iFrame	https://request.eprotect.vantivprelive.com/eProtect/js/eProtect-iframe-client.min.js
	POST Request Submission (for Mobile API)	https://request.eprotect.vantivprelive.com/eProtect/paypage
Post-Live (Regression Testing)	JavaScript Library	https://request.eprotect.vantivpostlive.com/eProtect/eProtect-api2.js
	Request Submission (excluding POST)	https://request.eprotect.vantivpostlive.com
	iFrame	https://request.eprotect.vantivpostlive.com/eProtect/js/eProtect-iframe-client.min.js
	POST Request Submission (for Mobile API)	https://request.eprotect.vantivprelive.com/eProtect/paypage
Live Production	*Production*	*Contact your Implementation Consultant for the eProtect Production URL.*

1.3.7.1 Pre-Live Environment Limitations and Maintenance Schedule

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

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

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

- Merchant profile and data is deleted after seven (7) consecutive days with no activity.
- Maintenance window - each Tuesday and Thursday from 4:00-8:00 AM ET.
- Daily limit of 1,000 Online transactions.
- Daily limit of 10,000 Batch transactions.

NOTE: **Due to the planned maintenance windows, you should not use this environment for continuous regression testing.**

1.3.8 Transitioning from Certification to Production

Before using your checkout form with eProtect in a production environment, replace all instances of the Testing and Certification URLs listed in Table 1-2 with the production URL. Contact Implementation for the appropriate production URL. **The URLs in Table 1-2 and in the sample scripts throughout this guide should only be used in the certification and testing environment.**

1.3.9 eProtect-Specific Response Codes

Table 1-3 and Table 1-4 list response codes specific to the eProtect feature, received in the browser or mobile device, and those received via a cnpAPI Response. For information on response codes specific to token transactions, see the *Vantiv cnpAPI Reference Guide*.

TABLE 1-3 eProtect-Specific Response Codes Received in Browsers or Mobile Devices

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

TABLE 1-3 eProtect-Specific Response Codes Received in Browsers or Mobile Devices

Response Code	Description	Error Type	Error Source
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
884	eProtect iFrame HTML failed to load	System	Vantiv eComm
885	eProtect iFrame CSS failed load - *<number>*	System	Vantiv eComm
889	Failure	System	Vantiv eComm

TABLE 1-4 eProtect Response Codes Received in cnpAPI Responses

Response Code	Response Message	Response Type	Description
877	Invalid PayPage Registration ID	Hard Decline	A eProtect response indicating that the Registration ID submitted is invalid.
878	Expired PayPage Registration ID	Hard Decline	A eProtect response indicating that the Registration ID has expired (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 eProtect feature.

1.4 Creating a Customized CSS for iFrame

Before you begin using the iFrame solution, you must create a Cascading Style Sheet (CSS) to customize the look and feel of the iFrame to match your checkout page. After creating and customizing your style sheet, you then submit the style sheet to Vantiv, where it will be tested before it is deployed into production. This section describes the various tools and customizations available for creating your CSS for iFrame and submitting your CSS for review:

- CSS iFrame Validation and Customization Features
- Using Web Developer Tools
- Reviewing your CSS with Vantiv

NOTE: If you are evaluating your styling options and/or having trouble creating your own style sheet, Vantiv can provide sample CSS files. Please contact your assigned Implementation Consultant for sample CSS files.

1.4.1 CSS iFrame Validation and Customization Features

Vantiv offers a set of iFrame validation and customization features to reduce cart abandonment, increase conversions, and help simplify the payment experience for your customers. See Configuring the iFrame on page 33 for further information on placement of these properties in your checkout page.

These features include:

Real-Time In-line Field Validations - while traditional web forms use submit-and-refresh rules that respond once you click the *Submit* button, real-time in-line validations can help your customers fill out web forms faster and with fewer errors. By guiding them with real-time feedback that instantly confirms whether the values they input are valid, transactions can be more successful and less error-prone, and customers are more satisfied with their checkout experience.

Payment Form Behaviors - customizable behaviors include:

- *Empty Input* - if your customer clicks back after leaving a payment form (for example, if they want to edit their payment information or in the case of a timeout, etc.), eProtect detects whether your customer has attempted to enter new form data.

  If they have not entered any new values, you can use the original token for the transaction. If your customer attempts to enter new values, eProtect clears the form—instead of leaving the previous entries in place—eliminating the need to erase the old values before re-entering new values.
- *Disallowed Characters* - allows only numeric values to be entered for the Account Number and CVC fields (no alpha or special characters are permitted).
- *Auto-Formatting* of account numbers based on the type of card.

Client Driven Behaviors - additional capabilities include:

- *Tooltips* - you can add a tool tip for any field (not just security code) activated by hovering, or when clicking 'What's This?'
- *Font Library and Icons* - Vantiv hosts SVG Icons (Font Awesome, v4.7.0) font library on our servers for you to leverage in your CSS, using an industry standard icon library for all icons.
- *Trust Badge* - you can add a 'trust' badge (e.g., a padlock or shield icon) on the payment form to reassure your customers that your site is legitimate and that all their personal data is collected securely through trusted third-party service providers. Note that the trust badge can be displayed *in place of* the card graphic; your page cannot display both.

Table 1-5, "iFrame Checkout Page Customizations - In-Line Field Validations" and Table 1-6, "Style Sheet and iFrame Customizations" show samples of these CSS iFrame customizations and describes the implementation of each.

When you set the optional `enhancedUxFeatures.inlineFieldValidations` property to `true` when configuring your iFrame, the behaviors listed in Table 1-5 are all included.

TABLE 1-5 iFrame Checkout Page Customizations - In-Line Field Validations

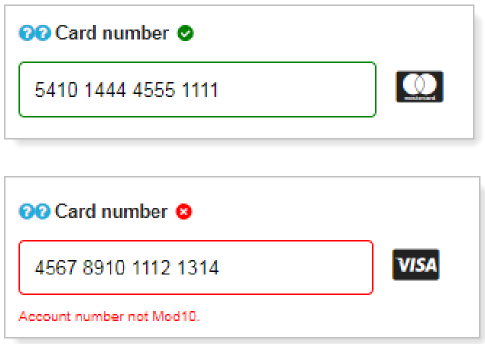
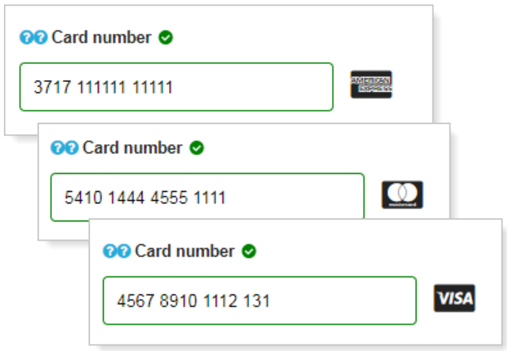
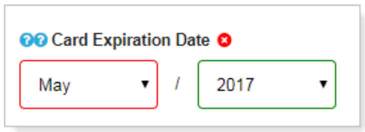
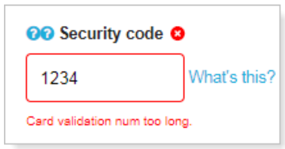
Field	Validation Behavior	Samples
Card Number	The iFrame checks the card number for correct size (too short or too long) and against the Luhn/Mod10 algorithm. In this example, if the consumer's inputs are valid, you can configure the iFrame to display green field borders and include a green check mark. Red borders and a red 'X' can indicate invalid input. The error messages and frame colors are customizable in your style sheet.	 Card number ✓ 5410 1444 4555 1111 Card number ✗ 4567 8910 1112 1314 Account number not Mod10.
	The iFrame identifies the card type (Visa, MasterCard, Amex, etc.) based on the first few digits entered, and displays the appropriate card graphic. If the card type is unknown, the iFrame displays a generic card graphic. You can configure your style sheet to hide the card graphic. In addition, the iFrame auto-formats the arrangement of the card digits based on the initial entry.	 Card number ✓ 3717 111111 11111 Card number ✓ 5410 1444 4555 1111 Card number ✓ 4567 8910 1112 131

TABLE 1-5 iFrame Checkout Page Customizations - In-Line Field Validations (Continued)

Field	Validation Behavior	Samples
Expiration Date	The iFrame checks the expiration month to determine if the selected month is prior to the current month.	 Card Expiration Date May / 2017
Security Code	The iFrame confirms the logic against the account number type. For example, if the card is an American Express card and the consumer enters only three digits (should be four digits), an error is indicated.	 Security code 1234 What's this? Card validation num too long.

The items listed in Table 1-6 are also available as optional features controlled by the your style sheet and via iFrame function. By default, the Tool Tip features are active, but can be suppressed with the CSS.

TABLE 1-6 Style Sheet and iFrame Customizations

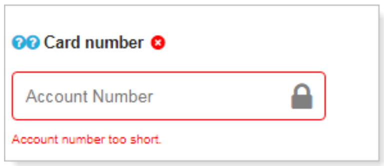
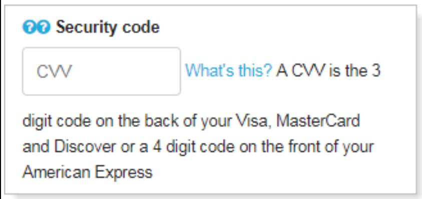
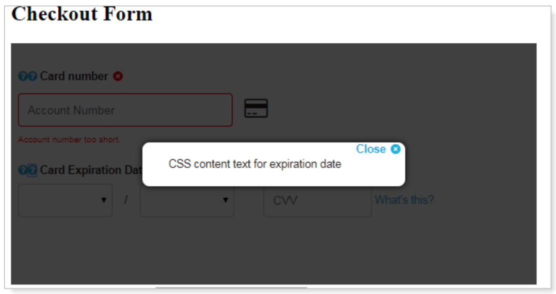
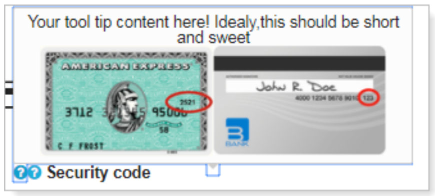
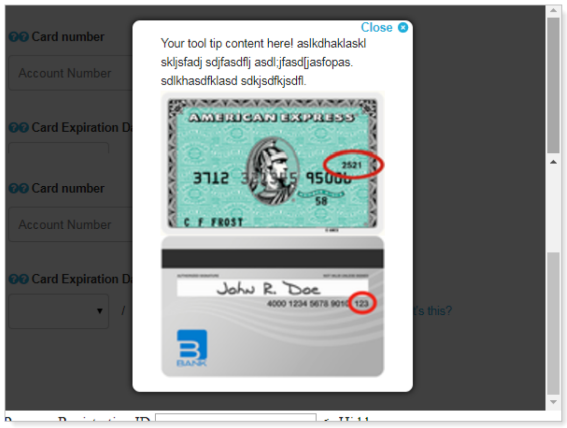
Customization	Samples
Trust Badge - You can add a 'trust badge' (e.g., a padlock or shield icon) to the payment form, using the Font Awesome (V4.7.0) icon library. Note that the trust badge can be displayed *in place of* the card graphic; your page cannot display both.	 Card number Account Number Account number too short.
Tool Tips - you control the following tool tip behavior in your style sheet:	
You can add a tool tip for any field (not just security code) activated by hovering, or when clicking 'What's This?'	 Security code CVV What's this? A CVV is the 3 digit code on the back of your Visa, MasterCard and Discover or a 4 digit code on the front of your American Express *Tool tip displayed after clicking 'What's This?'*

TABLE 1-6 Style Sheet and iFrame Customizations (Continued)

Customization	Samples
You can configure your style sheet to activate a tool tip by hovering over the '?' icon (rather than clicking). This is useful for short statements. You can also configure a modal dialog to activate on the click of the second '?' icon to display more lengthy CSS content.	 Checkout Form Card number Account Number Account number too short. Card Expiration Dat Close CSS content text for expiration date CVV What's this? *Modal dialog displayed upon clicking second '?' icon.*
Tool Tips (*continued*) You can configure your CSS to display a Security code modal dialog where the tool tip displays generic card art showing the placement of CVC on cards. You can hide this with the CSS, if you choose. You can also remove the scrollbars, as well as direct your CSS to auto-size the dialog based on content.	 Your tool tip content here! Idealy,this should be short and sweet Security code Card number Account Number Card Expiration D Card number Account Number Card Expiration D Close Your tool tip content here! aslkdhaklaskl skljsfadj sdjfasdflj asdl;jfasd[jasfopas. sdlkhasdfklasd sdkjsdfkjsdfl. 's this? *Modal dialog displayed upon clicking first or second '?' icon at the security code field.*

1.4.2 Using Web Developer Tools

By using standard browser-provided web developer tools, you can develop and customize your CSS prior to sending it to Vantiv for boarding.

To access the developer tool and to customize your CSS:

1. Go to https://www.testvantivcnp.com/iframe/ to access the demo URL and review the provided style sheet.

   If you are using the enhanced iFrame features described in the previous section, CSS iFrame Validation and Customization Features, use the following URL:

   https://www.testvantivcnp.com/checkout/checkout-iframe-enhanced-demo.jsp
2. Right click the **Account Number** text field, then click **Inspect** or **Inspect Element** (depending on your browser). The browser splits the window into two or more browser-specific developer frames.
3. Locate the highlighted HTML section in the developer tool frame of the browser where it shows `<input type="text" id="accountNumber"...`
4. Scroll up a few lines, and locate the HTML section, `<head>…</head>`. Expand the section with the arrow icon (if it is not already expanded).
5. Locate the HTML section `<style>…</style>`, which is the last child of the `<head/>` element, and expand it.
6. Double click the content, delete it, then paste in your new style sheet. To make the new CSS style effective, simply click somewhere else to exit the editing mode.
7. Copy and paste the CSS file and send it to your Vantiv Implementation Consultant for review.

1.4.3 Reviewing your CSS with Vantiv

Vantiv reviews your CSS by an automatic process which has white-listed allowed CSS properties and black-listed, 'dangerous' CSS values (such as URL, JavaScript, expression). Properties identified as such have been removed from the white list, and if used, will fail verification of the CSS. See Table B-24, "CSS Properties Excluded From the White List (not allowed)" for those properties not allowed.

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

Note the following:

- If additional properties and/or values are introduced in future CSS versions, those properties and values will be automatically black-listed until Vantiv can review and supplement the white-listed properties and values.
- Certain properties allow unacceptable values, including URL, JavaScript, or expression. This includes the **content** property, which allows you to enter 'Exp Date' instead of our provided

'Expiration Date' label. If the property contains a URL, JavaScript, expression, or `attr(href)`, Vantiv will fail verification of the CSS.

- Any property in the white list also allows its browser's extended values, where applicable.

See https://www.testvantivcnp.com/iframe/ to view a simple iFrame example.

To view an iFrame example checkout page using the enhanced features described in CSS iFrame Validation and Customization Features on page 14, use the following URL:

https://www.testvantivcnp.com/checkout/checkout-iframe-enhanced-demo.jsp

2

INTEGRATION AND TESTING

This chapter describes the steps required to integrate the eProtect™ feature as part of your checkout page, transaction examples, and information on eProtect testing and certification. The sections included are:

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

NOTE: **The PayPage product is now known as *Vantiv eProtect*. The term 'PayPage' however, is still used in this guide in certain text descriptions, along with many data elements, JS code, and URLs. Use of these data elements, etc., with the PayPage name is still valid with this release, but will transition to 'eProtect' in a future release.**

2.1 Integrating Customer Browser JavaScript API Into Your Checkout Page

This section provides step-by-step instructions for integrating the Customer Browser JavaScript API eProtect solution into your checkout page. This section also provides information on the foon Using the Customer Browser JavaScript API for Apple Pay on the Web

See Integrating eProtect Into Your Mobile Application on page 39 for more information on the mobile solution.

See Integrating iFrame into your Checkout Page on page 32 for more information on the iFrame solution.

2.1.1 Integration Steps

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

1. Loading the eProtect API and jQuery
2. Specifying the eProtect API Request Fields
3. Specifying the eProtect 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 eProtect API

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

```
sendToEprotect(eProtectRequest, eProtectFormFields, successCallback, errorCallback,
timeoutCallback, timeout)
```

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

JavaScript code examples are included with each step. For a full HTML code example of the eProtect implementation, see the HTML Checkout Page Examples on page 74.

2.1.2 Loading the eProtect API and jQuery

To load the eProtect client JavaScript library from the eProtect 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 eProtect client JavaScript library.

NOTE: **To avoid disruption to transaction processing, Vantiv recommends you download the latest JavaScript client to your checkout page a minimum of once per day (due to frequent changes to the JavaScript client). Vantiv does not recommend caching the eProtect JavaScript client on your servers.**

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

```
<head>
...
<script
   src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
type="text/javascript">
  </script>
<script
   src="https://request.eprotect.vantivprelive.com/eProtect/eProtect-api2.js"
type="text/javascript">
  </script>
...
</head>
```

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

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 eProtect in a production environment, replace the certification URL with the production URL (contact your Implementation Consultant for the appropriate production URL).**

2.1.3 Specifying the eProtect API Request Fields

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

NOTE: **The `orderId` field must be a text string with a maximum of 25 characters. The values for either the `merchantTxnId` or the `orderId` must be unique so that we can use these identifiers for reconciliation or troubleshooting.**

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

```
<form
  <input type="text" id="ccNum" size="20">
  <input type="text" id="cvv2Num" size="4">
  <input type="text" id="paypageRegistrationId" name="paypageRegistrationId"
readonly="true" hidden>
  <input type="text" id="bin" name="bin" readonly="true" hidden>
  <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>
```

TABLE 2-1 `eProtectFormFields` Definitions

Field	Description
`ccNum`	(*Optional*) The credit card account number.
`cvv2Num`	(*Optional*) The card validation number, either the CVV2 (Visa), CVC2 (MasterCard), or CID (American Express and Discover) value.
`paypageRegistrationId`	(*Required*) The temporary identifier used to facilitate the mapping of a token to a card number.
`bin`	(*Optional*) The bank identification number (BIN), which is the first six digits of the credit card number.

2.1.4 Specifying the eProtect API Response Fields

To specify the eProtect API Response fields, add seven hidden response fields on your checkout form for storing information returned by eProtect: `paypageRegistrationId`, `bin`, `code`, `message`, `responseTime`, `type`, and `vantivTxnId`. You have 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$vantivTxnId" name="response$vantivTxnId"
readOnly="true"/>
  <input type="hidden" id="response$firstSix" name="response$firstSix"
readOnly="true"/>
  <input type="hidden" id="response$lastFour" name="response$lastFour"
readOnly="true"/>
  ...
</form>
```

2.1.5 Handling the Mouse Click

In order to call the eProtect 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 eProtect Request and calls `sendToEprotect`.

The `sendToEprotect` call includes a timeout value in milliseconds. We recommend a timeout value of 15000 (15 seconds). This value is based on data that only 1% of traffic exceeds five seconds. If you set your timeout value at 5000 (five seconds), we recommend that you follow up with a longer 15-second timeout value.

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

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

```
        var eProtectRequest = {
         "paypageId" : document.getElementById("request$paypageId").value,
         "reportGroup" : document.getElementById("request$reportGroup").value,
         "orderId" : document.getElementById("request$orderId").value,
         "id" : document.getElementById("request$merchantTxnId").value,
         "applepay" : applepay
         "url" : "https://request.eprotect.vantivprelive.com"

        };
        new eProtect().sendToEprotect(eProtectRequest, formFields, submitAfterEprotect,
onErrorAfterEprotect, timeoutOnEprotect, 15000);
              return false;
  ...
  </script>
...
</head>
```

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

TABLE 2-2 `eProtectRequest` Fields

Field	Description
paypageId	(*Required*) The unique number assigned by Implementation.
reportGroup	(*Required*) The cnpAPI required attribute that defines under which merchant sub-group this transaction will be displayed in eCommerce iQ Reporting and Analytics.
orderId	The merchant-assigned unique value representing the order in your system (used when linking authorizations, captures, and refunds, and for retries). Vantiv recommends that the values for `id` and `orderId` be different and unique so that we can use these identifiers for reconciliation or troubleshooting. If you do not have the order number available at this time, please generate another unique number to send as the `orderId` (and send it to your servers to map it to the order number that you generate later).
id	The merchant-assigned unique value representing this transaction in your system. The same value must be used for retries of the same failed eProtect transaction but must be unique between the eProtect transaction, authorization, capture, and refund for the same order. Vantiv recommends that the values for `id` and `orderId` must be different and unique so that we can use these identifiers for reconciliation or troubleshooting.
applepay	(*Optional*). The Apple Pay PKPaymentToken. Required for Apple Pay on the Web.
url	(*Required*) The URL to request submission for eProtect. See Table 1-2, eProtect Certification, Testing, and Production URLs on page 11.

2.1.6 Intercepting the Checkout Form Submission

Without the eProtect implementation, order data is sent to your system when the submit button is clicked. With the eProtect feature, a request must be sent to our server to retrieve the 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`.

Note that this also serves as a method for detecting JavaScript and informing customers that JavaScript must be enabled in this checkout process.

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

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

2.1.7 Handling Callbacks for Success, Failure, and Timeout

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

2.1.7.1 Success Callbacks

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

```
<head>
  ...
  <script>
  ...
function setEprotectResponseFields(response) {
  document.getElementById('response$code').value = response.response;
  document.getElementById('response$message').value = response.message;
  document.getElementById('response$responseTime').value = response.responseTime;
  document.getElementById('response$vantivTxnId').value = response.vantivTxnId;
  document.getElementById('response$type').value = response.type;
  document.getElementById('response$firstSix').value = response.firstSix;
  document.getElementById('response$lastFour').value = response.lastFour;
}
function submitAfterEprotect (response) {
  setEprotectResponseFields(response);
  document.forms['fCheckout'].submit();
 }
  ...
  </script>
 ...
</head>
```

2.1.7.2 Failure Callbacks

There are two types of failures that can occur when your customer enters an order: validation (user) errors, and system (non-user) errors (see Table 1-3, "eProtect-Specific Response Codes Received in Browsers or Mobile Devices" on page 12). 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 eProtect Integration and also have enough information for debugging.**

You have flexibility in the wording of the error text.

```
<head>
...
   <script>
   ...
   function onErrorAfterEprotect (response) {
     setEprotectResponseFields(response);
     if(response.response == '871') {
       alert("Invalid card number.  Check and retry. (Not Mod10)");
     }
     else if(response.response == '872') {
       alert("Invalid card number.  Check and retry. (Too short)");
     }
     else if(response.response == '873') {
       alert("Invalid card number.  Check and retry. (Too long)");
     }
     else if(response.response == '874') {
       alert("Invalid card number.  Check and retry. (Not a number)");
     }
     else if(response.response == '875') {
       alert("We are experiencing technical difficulties. Please try again later or call
555-555-1212");
     }
     else if(response.response == '876') {
       alert("Invalid card number.  Check and retry. (Failure from Server)");
     }
     else if(response.response == '881') {
          alert("Invalid card validation code.  Check and retry. (Not a number)");
        }
        else if(response.response == '882') {
          alert("Invalid card validation code.  Check and retry. (Too short)");
        }
        else if(response.response == '883') {
          alert("Invalid card validation code.  Check and retry. (Too long)");
        }
        else if(response.response == '889') {
          alert("We are experiencing technical difficulties. Please try again later or call
555-555-1212");
      }
 return false;
}
 ...
 </script>
...
</head>
```

2.1.7.3 Timeout Callbacks

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

Timeout values are expressed in milliseconds and defined in the `sendToEprotect` call, described in the section, Handling the Mouse Click on page 25. We recommend a timeout value of 15000 (15 seconds).

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

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

2.1.8 Detecting the Availability of the eProtect API

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

```
</BODY>
...
<script>
```

```
    function callEprotect() {
      if(typeof eProtect !== 'function') {
        alert("We are experiencing technical difficulties.  Please try again later or
call 555-555-1212 (API unavailable)" );
</script>
...
</HTML>
```

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

2.1.9 Using the Customer Browser JavaScript API for Apple Pay on the Web

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

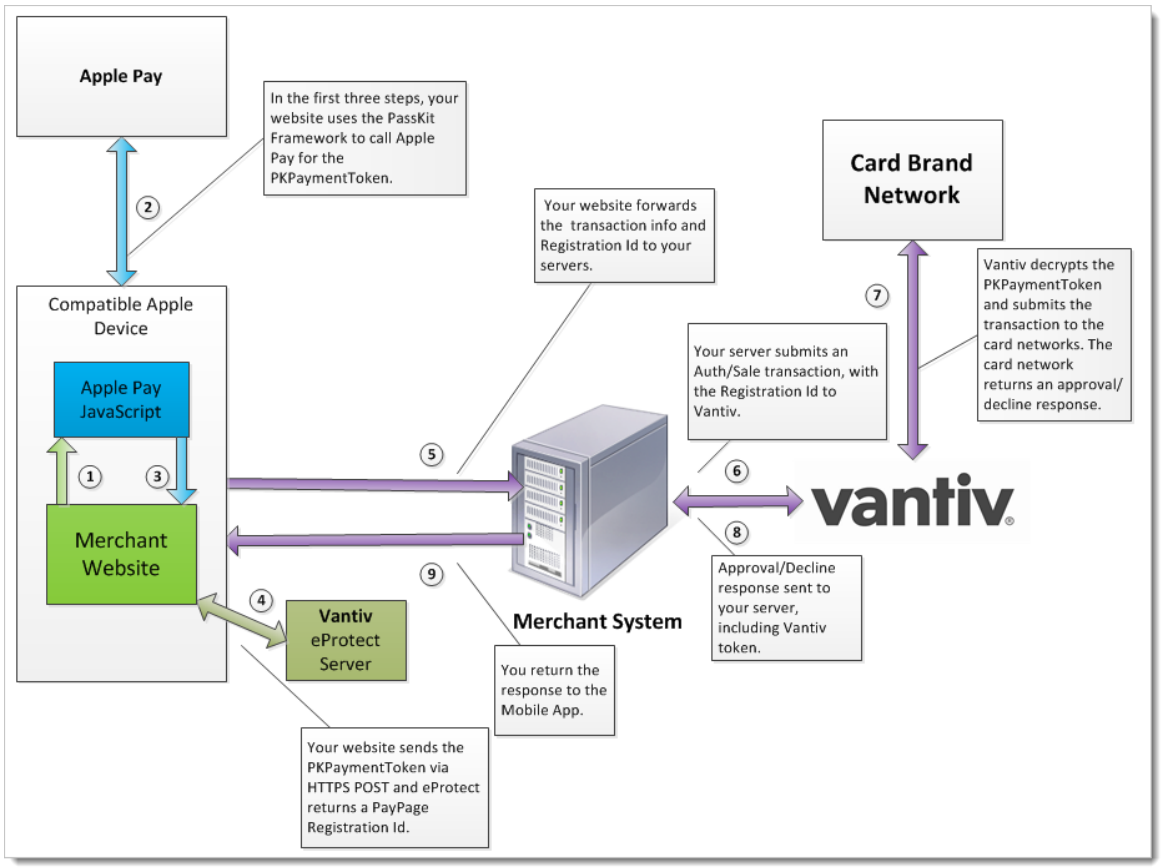
In this scenario, the Vantiv eProtect Customer Browser JavaScript API controls the fields on your checkout page that hold sensitive card data. When the cardholder clicks the Apple Pay button, communication is exchanged with Apple Pay via the JavaScript API to obtain the PKPaymentToken. From this point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

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

1. When the consumer selects the Apple Pay option from your website, your site makes use of the Apple Pay JavaScript to request payment data from Apple Pay.
2. When Apple Pay receives the call from your website and after the consumer approves the Payment Sheet (using Touch ID), Apple creates a PKPaymentToken using your public key. Included in the PKPaymentToken is a network (Visa, MasterCard, American Express, or Discover) payment token and a cryptogram.
3. Apple Pay returns the Apple PKPaymentToken (defined in Apple documentation; please refer to https://developer.apple.com/library/content/documentation/PassKit/Reference/PaymentTokenJSON/PaymentTokenJSON.html) to your website.
4. Your website sends the PKPaymentToken to our secure server via the JavaScript Browser API and eProtect returns a Registration ID.
5. Your website forwards the transaction data along with the Registration ID to your order processing server, as it would with any eProtect transaction.

6. Your server constructs/submits a standard cnpAPI Authorization/Sale transaction using the Registration ID, setting the `<orderSource>` element to `applepay`.
7. Using the private key, Vantiv decrypts the PKPaymentToken associated with the Registration ID and submits the transaction with the appropriate information to the card networks for approval.
8. Vantiv sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response and includes the Vantiv token.
9. You return the Approval/Decline message to your website.

FIGURE 2-1 Data/Transaction Flow - Customer Browser JavaScript API for Apple Pay Web


Apple Pay
In the first three steps, your website uses the PassKit Framework to call Apple Pay for the PKPaymentToken.
2
Card Brand Network
Your website forwards the transaction info and Registration Id to your servers.
Compatible Apple Device
Apple Pay JavaScript
7
Vantiv decrypts the PKPaymentToken and submits the transaction to the card networks. The card network returns an approval/ decline response.
Your server submits an Auth/Sale transaction, with the Registration Id to Vantiv.
1
3
5
6
Merchant Website
vantiv
8
9
4
Vantiv eProtect Server
Merchant System
Approval/Decline response sent to your server, including Vantiv token.
You return the response to the Mobile App.
Your website sends the PKPaymentToken via HTTPS POST and eProtect returns a PayPage Registration Id.

2.2 Integrating iFrame into your Checkout Page

This section provides information and instructions for integrating the iFrame eProtect solution into your checkout page. Review the section Creating a Customized CSS for iFrame on page 14 for information on creating a style sheet. Also see https://www.testvantivcnp.com/iframe/ to view our iFrame example page.

2.2.1 Integration Steps

Integrating the iFrame into you checkout page includes the following steps, described in the sections to follow. For a full HTML code example a iFrame eProtect implementation, see the HTML Example for Hosted iFrame-Integrated Checkout Page on page 78.

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

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

2.2.2 Loading the iFrame

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

```
<script src="https://request.eprotect.vantivprelive.com/eProtect/js/eProtect-iframe-
client.min.js"></script>
```

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

2.2.3 Configuring the iFrame

To configure the iFrame after the page is loaded, you specify the required properties listed in Table 2-3 (other properties shown in the example below, are optional). You define a callback for errors, time-outs, and to retrieve the `paypageRegistrationId`. In this example, this is called `eProtectiframeClientCallback`.

```
$( document ).ready(function() {
  var configure = {
    "paypageId":document.getElementById("request$paypageId").value,
    "style":"test",
    "reportGroup":document.getElementById("request$reportGroup").value,
    "timeout":document.getElementById("request$timeout").value,
    "div": "eProtectiframe",
    "callback": eProtectiframeClientCallback,
    "showCvv": true,
    "months": {
      "1":"January",
      "2":"February",
      "3":"March",
      "4":"April",
      "5":"May",
      "6":"June",
      "7":"July",
      "8":"August",
      "9":"September",
      "10":"October",
      "11":"November",
      "12":"December"
    },
    "numYears": 8,
    "tooltipText": "A CVV is the 3 digit code on the back of your Visa, MasterCard and Discover or a 
4 digit code on the front of your American Express",
    "tabIndex": {
      "cvv":1,
      "accountNumber":2,
      "expMonth":3,
      "expYear":4
    },
    "placeholderText": {
      "cvv":"CVV",
      "accountNumber":"Account Number"
    },
        "inputsEmptyCallback": inputsEmptyCallback,
        "enhancedUxFeatures" : {
        "inlineFieldValidations": true,
        }

  };
  if(typeof eProtectiframeClient === 'undefined') {
      alert("We are experiencing technical difficulties.  Please try again or call us to complete 
your order");
  //You may also want to submit information you have about the consumer to your servers to 
facilitate debugging like customer ip address, user agent, and time
  }

  else {
  var eProtectiframeClient = new EprotectIframeClient(configure);
  eProtectiframeClient.autoAdjustHeight();

});
```

TABLE 2-3 Common Properties

Property	Description
`paypageId`	(*Required*) The unique number assigned by Implementation.
`style`	(*Required*) The CSS filename (excluding the '.css'). For example, if the style sheet filename is `mysheet1.css`, the value for this property is `mysheet1`.
`reportGroup`	(*Required*) The cnpAPI required attribute that defines under which merchant sub-group this transaction will be displayed in eCommerce iQ Reporting and Analytics.
`timeout`	(*Required*) The number of milliseconds before a transaction times out and the timeout callback in invoked. Vantiv recommends a timeout value of 15000 (15 seconds). This value is based on data that only 1% of traffic exceeds five seconds. If you set your timeout value at 5000 (five seconds), we recommend that you follow up with a longer 15-second timeout value.
`div`	(*Required*) The ID of the HTML `div` element where our iFrame is embedded as innerHTML.

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

`inputsEmptyCallback`	*(Optional)* When a consumer returns to your checkout page to edit non-payment information, this function determines whether the Card number and security code fields are empty, and indicates whether to return this information in your callback. See Creating a Customized CSS for iFrame *on page 14* for more information.
`inlineFieldValidations`	*(Optional)* Determines whether in-field validations are performed (set value to `true`). See Creating a Customized CSS for iFrame *on page 14* for more information.

TABLE 2-3 Common Properties (Continued)

Property	Description
`height`	(*Optional*) The height (in pixels) of the iFrame. There are three options: • You can pass `height` as an optional parameter when configuring the client. • You can call `autoAdjustHeight` in the client to tell the iFrame to adjust the height to exactly the number of pixels needed to display everything in the iFrame without displaying a vertical scroll bar (recommended). • You can ignore `height`. The iFrame may display a vertical scroll bar, depending upon your styling of the `div` containing the iFrame.
`htmlTimeout`	(*Optional*) The amount of time (in milliseconds) to wait for the iFrame to load before responding with an '884' error code.

2.2.4 Calling the iFrame for the Registration ID

After your customer clicks the Submit/Complete Order button, your checkout page must call the iFrame to get a Registration ID. In the `onsubmit` event handler of your button, add code to call eProtect to get a Registration ID for the account number and CVV. Include the parameters listed in Table 2-4.

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

TABLE 2-4 Event Handler Parameters

Parameter	Description
`id`	The merchant-assigned unique value representing this transaction in your system. The same value must be used for retries of the same failed eProtect transaction but must be unique between the eProtect transaction, authorization, capture, and refund for the same order. **Type**: String **Max Length**: 25 characters Vantiv recommends that the values for `id` and `orderId` must be different and unique so that we can use these identifiers for reconciliation or troubleshooting.
`orderId`	The merchant-assigned unique value representing the order in your system (used when linking authorizations, captures, and refunds, and for retries). **Type**: String **Max Length**: 25 characters Vantiv recommends that the values for `id` and `orderId` be different and unique so that we can use these identifiers for reconciliation or troubleshooting. If you do not have the order number available at this time, please generate another unique number to send as the `orderId` (and send it to your servers to map it to the order number that you generate later).

2.2.5 Handling Callbacks

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

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

```
var eProtectiframeClientCallback = function(response) {
  if (response.timeout) {
  alert("We are experiencing technical difficulties.  Please try again or call us to complete your 
order");
```

```
    //You may also want to submit information you have about the consumer to your servers to
facilitate debugging like customer ip address, user agent, and time
  }
  else {
    document.getElementById('response$code').value = response.response;
    document.getElementById('response$message').value = response.message;
    document.getElementById('response$responseTime').value = response.responseTime;
    document.getElementById('response$reportGroup').value = response.reportGroup;
    document.getElementById('response$merchantTxnId').value = response.id;
    document.getElementById('response$orderId').value = response.orderId;
    document.getElementById('response$vantivTxnId').value = response.vantivTxnId;
    document.getElementById('response$type').value = response.type;
    document.getElementById('response$lastFour').value = response.lastFour;
    document.getElementById('response$firstSix').value = response.firstSix;
    document.getElementById('paypageRegistrationId').value = response.paypageRegistrationId;
    document.getElementById('bin').value = response.bin;
    document.getElementById('response$expMonth').value = response.expMonth;
    document.getElementById('response$expYear').value = response.expYear;
    if(response.response === '870') {
      //Submit the form
    }
    else if(response.response === '871' || response.response === '872' || response.response ===
'873' || response.response === '874' || response.response === '876') {
      //Recoverable error caused by user mis-typing their credit card
        alert("Please check and re-enter your credit card number and try again.");
    }
    else if(response.response === '881' || response.response === '882' || response.response === 883)
{
      //Recoverable error caused by user mis-typing their credit card
        alert("Please check and re-enter your card validation number and try again.");
    }
    else if(response.response === '884') {
      //Frame failed to load, so payment can't proceed.
      //You may want to consider a larger timeout value for the htmlTimeout property
      //You may also want to log the customer ip, user agent, time, paypageId and style that failed
to load for debugging.
      //Here, we hide the frame to remove the unsightly browser error message from the middle of
our payment page that may eventually display
      $('#eProtectiframe').hide();
      // and disable the checkout button
      $('#submitButton').attr('disabled','disabled');
    }
    else if(response.response === '885') {
      //CSS Failed to load, so the page will look unsightly but will function.
      //We are going to continue with the order
      $('#submitButton').removeAttr('disabled');
      //You may also want to log the customer ip, user agent, time, and style that failed to load
for debugging
    }

    else {
      //Non-recoverable or unknown error code
        alert("We are experiencing technical difficulties. Please try again or call us to complete
your order");
      //You may also want to submit the vantivTxnId and response received, plus information you
have about the consumer to your servers to facilitate debugging, i.e., customer ip address, user
agent and time
        }
      }
    };
```

2.2.5.1 Handling Errors

In case of errors in the iFrame, the iFrame adds an error class to the field that had the error. You can use those classes in the CSS you give Vantiv Implementation to provide error styles. The codes correspond to the response codes outlined in eProtect-Specific Response Codes on page 12.

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

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

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

2.3 Integrating eProtect Into Your Mobile Application

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

- Using the Vantiv Mobile API for Apple Pay
- Using the Vantiv Mobile API for Android Pay
-

2.3.1 Creating the POST Request

You structure your POST request as shown in the Sample Request. Use the components listed in Table 2-5. The URLs and User Agent examples in this table (in red) should only be used in the certification and testing environment. For more information on the appropriate User Agent (iOS and Android versions can differ), see the HTTP standard at http://www.ietf.org/rfc/rfc2616.txt section 14.43.

TABLE 2-5 POST Headers, Parameters, and URL

Component	Element	Description
Headers (optional)	`Content-Type: application/x-www-form-urlencoded`	
	`Host: request.eprotect.vantivprelive.com`	
	`User-Agent = "User-Agent" ":" 1*( product | comment )` *For example:* `User-Agent: Vantiv/1.0 CFNetwork/459 Darwin/10.0.0.d3`	

TABLE 2-5 POST Headers, Parameters, and URL

Component	Element	Description
Parameters (required)	**paypageId**	The unique number assigned by Implementation.
	reportGroup	The cnpAPI-required attribute that defines under which merchant sub-group this transaction will be displayed in eCommerce iQ Reporting and Analytics.
	orderId	A unique value that you assign (string, max length: 25 char.). See full definition on page 26.
	id	A unique value that you assign (string, max length: 25 char.). See full definition on on page 26.
	accountNumber	The 13-25-digit card account number. (Not used in Apple Pay transactions.)
(optional)	**cvv**	The card validation number, either the CVV2 (Visa), CVC2 (MasterCard), or CID (American Express and Discover) value. (Not used in Apple Pay transactions.)
URL	**https://request.eprotect.vantivprelive.com/eProtect/paypage**	

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

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 eProtect in a production environment, replace the certification URL with the production URL (contact your Implementation Consultant for the appropriate production URL).**

2.3.1.1 Sample Request

The following is an example POST to request a Registration ID:

```
$ curl --verbose -H "Content-Type: application/x-www-form-urlencoded" -H "Host:
request.eprotect.vantivprelive.com/eProtect/paypage" -H "User-Agent: Vantiv/1.0
CFNetwork/459 Darwin/10.0.0.d3" -d"paypageId=a2y4o6m8k0&
reportGroup=*merchant1500&orderId=PValid&id=12345&accountNumber=ACCOUNT_NUMBER&cvv=CVV
"https://request.eprotect.vantivprelive.com/eProtect/paypage
```

2.3.1.2 Sample Response

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

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

Table 2-6 lists the parameters included in the response.

TABLE 2-6 Parameters Returned in POST Response

Parameter	Description
`bin`	The bank identification number (BIN), which is the first six digits of the credit card number
`firstSix`	(Mirrored back from the request) The first six digits of the credit card number.
`lastFour`	(Mirrored back from the request) The last four digits of the credit card number.
`paypageRegistrationId`	The temporary identifier used to facilitate the mapping of a token to a card number.
`type`	The method of payment for this transaction (VI=Visa, MC=MasterCard, AX=Amex, DI=Discover).
`id`	(Mirrored back from the request) The merchant-assigned unique value representing this transaction in your system. **Type**: String **Max Length**: 25 characters
`vantivTxnId`	The automatically-assigned unique transaction identifier.
`message`	The transaction response returned by Vantiv, corresponding to the `response` reason code. If the transaction was declined, this message provides a reason.
`orderId`	(Mirrored back from the request) The merchant-assigned unique value representing the order in your system. **Type**: String **Max Length**: 25 characters
`reportGroup`	(Mirrored back from the request) The cnpAPI required attribute that defines under which merchant sub-group this transaction will be displayed in eCommerce iQ Reporting and Analytics.
`response`	The three-digit transaction response code returned by Vantiv for this transaction.
`responseTime`	The date and time (GMT) the transaction was processed.

2.3.2 Using the Vantiv Mobile API for Apple Pay

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

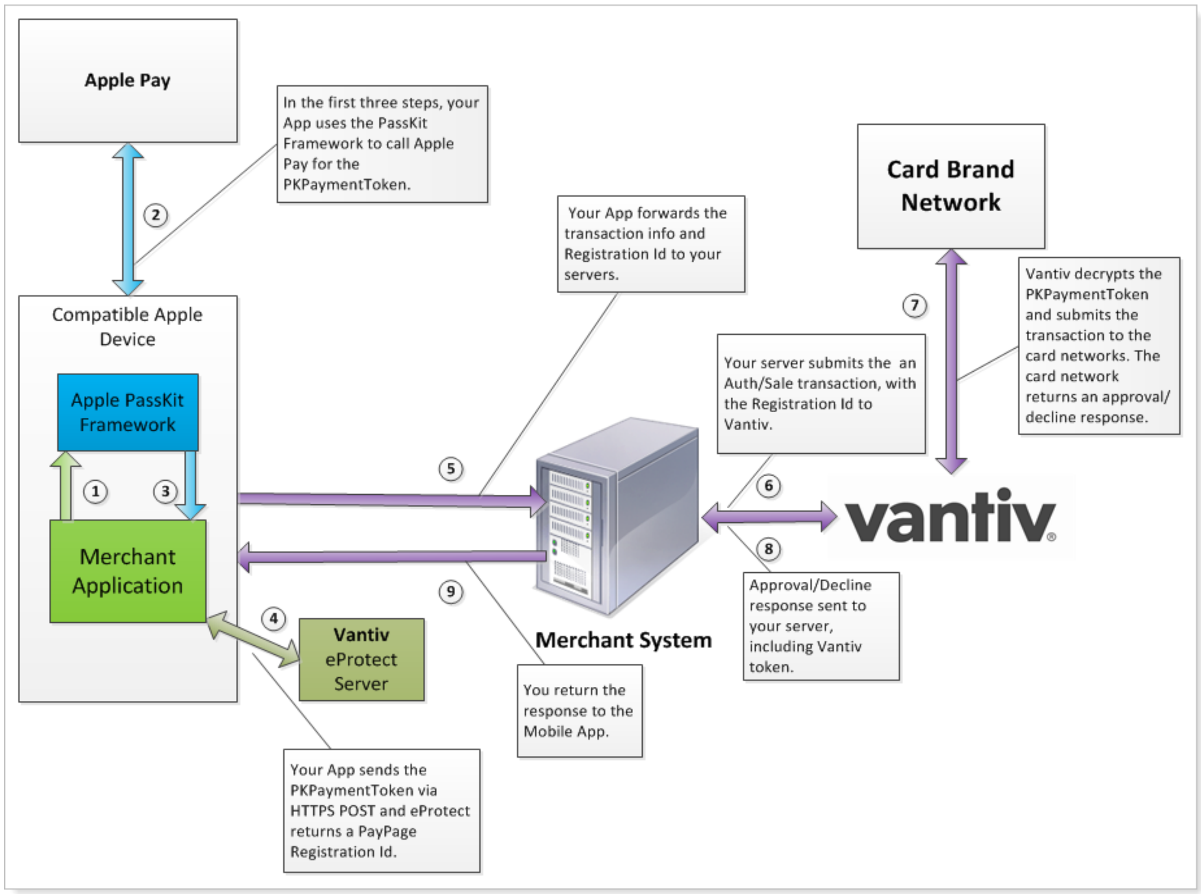
In this scenario, your native iOS application performs an HTTPS POST of the Apple Pay PKPaymentToken using the Vantiv Mobile API for Apple Pay. From this point forward, your handling of the transaction is identical to any other eProtect transaction. The eProtect server returns a Registration ID and your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

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

1. When the consumer selects the Apple Pay option from your application or website, your application/site makes use of the Apple PassKit Framework to request payment data from Apple Pay.
2. When Apple Pay receives the call from your application or website and after the consumer approves the Payment Sheet (using Touch ID), Apple creates a PKPaymentToken using your public key. Included in the PKPaymentToken is a network (Visa, MasterCard, American Express, or Discover) payment token and a cryptogram.
3. Apple Pay returns the Apple PKPaymentToken (defined in Apple documentation; please refer to https://developer.apple.com/library/content/documentation/PassKit/Reference/PaymentTokenJSON/PaymentTokenJSON.html) to your application.
4. Your native iOS application sends the PKPaymentToken to our secure server via an HTTPS POST (see Creating a POST Request for an Apple Pay Transaction on page 43) and eProtect returns a Registration ID.
5. Your native iOS application forwards the transaction data along with the Registration ID to your order processing server, as it would with any eProtect transaction.
6. Your server constructs/submits a standard cnpAPI Authorization/Sale transaction using the Registration ID.
7. Using the private key, Vantiv decrypts the PKPaymentToken associated with the Registration ID and submits the transaction with the appropriate information to the card networks for approval.
8. Vantiv sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response and includes the Vantiv token.
9. You return the Approval/Decline message to your mobile application.

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

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


Apple Pay
In the first three steps, your App uses the PassKit Framework to call Apple Pay for the PKPaymentToken.
2
Compatible Apple Device
Apple PassKit Framework
1
3
Merchant Application
4
Vantiv eProtect Server
Your App sends the PKPaymentToken via HTTPS POST and eProtect returns a PayPage Registration Id.
Your App forwards the transaction info and Registration Id to your servers.
5
9
Merchant System
You return the response to the Mobile App.
Card Brand Network
7
Vantiv decrypts the PKPaymentToken and submits the transaction to the card networks. The card network returns an approval/decline response.
Your server submits the an Auth/Sale transaction, with the Registration Id to Vantiv.
6
vantiv
8
Approval/Decline response sent to your server, including Vantiv token.

2.3.2.1 Creating a POST Request for an Apple Pay Transaction

Construct your HTTPS POST as detailed in Creating the POST Request on page 39, using the components listed in the Table 2-5 as well as those listed in Table 2-7 (all required). See the Sample Apple Pay POST Request and Sample Apple Pay POST Response below.

TABLE 2-7 Vantiv Mobile API for Apple Pay HTTPS POST Required Components

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

2.3.2.2 Sample Apple Pay POST Request

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

```
curl --verbose -H "Content-Type: application/x-www-form-urlencoded" -H "Host:MerchantApp"
-H "User-Agent:Vantiv/1.0 CFNetwork/459 Darwin/10.0.0.d3"
-d"paypageId=a2y4o6m8k0&reportGroup=*merchant1500&orderId=PValid&id=1234&applepay.data=HT
897mACd%2F%2FTpWe10A5y9RmL5UfboTiDIvjni3zWFtyy8dtv72RJL1bk%2FU4dTDlrq1T1V2l0TSnI%0APLdOnn
HBO51bt9Ztj9odDTQ5LD%2F4hMZTQj3lBRvFOtTtjp9ysBAsydgjEjcCcbnkx7dCqgnwgzuz%0Ay7bX%2B5Fo8a8R
KqoprkDPwIMWOC9yWe7MQw%2FboM5NY2QtIcIvzbLFcYUxndYTg0IXNBHNzsvUOjmw%0AvEnMhXxeCH%2BC4KoC6M
EsAGK5rH1T5dSvTZzHF5c12dpsqdI73%2FBk6qEcdlT7gJKVmyDQC%2FNFxJ0X%0AF993Of6ejQDJq6BZsz8X7kYC
yJdI%2FPFJPZp4e3L%2FtCsBDUTJAgFLt2xF8HWaPoW8psILOGCCvJQm%0ATR1m7ODtSChaWOb7eYm1BpNiD3wkCH
8nmIMrlnt3KP4SeQ%3D%3D&applepay.signature=MIAGCSqGSIb3DQEHAqCAMIACAQExDzANBglghkgBZQMEAgE
FADCABgkqhkiG9w0BBwEAAKCAMIICvzCCAmWgAwIBAgIIQpCV6UIIb4owCgYIKoZIzj0EAwIwejEuMCwGA1UEAwwl
QXBwbGUgQXBwbGljYXRpb24gSW50ZWdyYXRpb24gQ0EgLSBHMzEmMCQGA1UECwwdQXBwbGUgQ2VydGlmaWNhdGlvb
iBBdXRob3JpdHkxEzARBgNVBAoMCkFwcGxlIEluYy4xCzAJBgNVBAYTAlVTMB4XDTE0MDUwODAxMjMzOVoXDTE5MD
UwNzAxMjMzOVowXzElMCMGA1UEAwwcZWNjLXNtcC1icm9rZXItc2lnbl9VQzQtUFJPRDEUMBIGA1UECwwLaU9TIFN
5c3RlbXMxEzARBgNVBAoMCkFwcGxlIEluYy4xCzAJBgNVBAYTAlVTMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE
whV37evWx7Ihj2jdcJChIY3HsL1vLCg9hGCV2Ur0pUEbg0IO2BHzQH6DMx8cVMP36zIg1rrV1O%2F0komJPnwPE6O
B7zCB7DBFBggrBgEFBQcBAQQ5MDcwNQYIKwYBBQUHMAGGKWh0dHA6Ly9vY3NwLmFwcGxlLmNvbS9vY3NwMDQtYXBw
bGVhaWNhMzAxMB0GA1UdDgQWBBSUV9tv1XSBhomJdi9%2BV4UH55tYJDAMBgNVHRMBAf8EAjAAMB8GA1UdIwQYMBa
AFCPyScRPk%2BTvJ%2BbE9ihsP6K7%2FS5LMDQGA1UdHwQtMCswKaAnoCWGI2h0dHA6Ly9jcmwuYXBwbGUuY29tL2
```

```
FwcGxlYWljYTMuY3JsMA4GA1UdDwEB%2FwQEAwIHgDAPBgkqhkiG92NkBh0EAgUAMAoGCCqGSM49BAMCA0gAMEUCI
QCFGdtAk%2B7wXrBV7jTwzCBLE%2BOcrVL15hjif0reLJiPGgIgXGHYYeXwrn02Zwcl5TT1W8rIqK0QuIvOnO1THC
bkhVowggLuMIICdaADAgECAghJbS%2B%2FOpjalzAKBggqhkjOPQQDAjBnMRswGQYDVQQDDBJBcHBsZSBSb290IEN
BIC0gRzMxJjAkBgNVBAsMHUFwcGxlIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MRMwEQYDVQQKDApBcHBsZSBJbmMu
MQswCQYDVQQGEwJVUzAeFw0xNDA1MDYyMzQ2MzBaFw0yOTA1MDYyMzQ2MzBaMHoxLjAsBgNVBAMMJUFwcGxlIEFwc
GxpY2F0aW9uIEludGVncmF0aW9uIENBIC0gRzMxJjAkBgNVBAsMHUFwcGxlIENlcnRpZmljYXRpb24gQXV0aG9yaX
R5MRMwEQYDVQQKDApBcHBsZSBJbmMuMQswCQYDVQQGEwJVUzBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABPAXEYQ
Z12SF1RpeJYEHduiAou%2Fee65N4I38S5PhM1bVZls1riLQl3YNIk57ugj9dhfOiMt2u2ZwvsjoKYT%2FVEWjgfcw
gfQwRgYIKwYBBQUHAQEEOjA4MDYGCCsGAQUFBzABhipodHRwOi8vb2NzcC5hcHBsZS5jb20vb2NzcDA0LWFwcGxlc
m9vdGNhZzMwHQYDVR0OBBYEFCPyScRPk%2BTvJ%2BbE9ihsP6K7%2FS5LMA8GA1UdEwEB%2FwQFMAMBAf8wHwYDVR
0jBBgwFoAUu7DeoVgziJqkipnevr3rr9rLJKswNwYDVR0fBDAwLjAsoCqgKIYmaHR0cDovL2NybC5hcHBsZS5jb20
vYXBwbGVyb290Y2FnMy5jcmwwDgYDVR0PAQH%2FBAQDAgEGMBAGCiqGSIb3Y2QGAg4EAgUAMAoGCCqGSM49BAMCA2
cAMGQCMDrPcoNRFpmxhvs1w1bKYr%2F0F%2B3ZD3VNoo6%2B8ZyBXkK3ifiY95tZn5jVQQ2PnenC%2FgIwMi3VRCG
wowV3bF3zODuQZ%2F0XfCwhbZZPxnJpghJvVPh6fRuZy5sJiSFhBpkPCZIdAAAxggFfMIIBWwIBATCBhjB6MS4wLA
YDVQQDDCVBcHBsZSBBcHBsaWNhdGlvbiBJbnRlZ3JhdGlvbiBDQSAtIEczMSYwJAYDVQQLDB1BcHBsZSBDZXJ0aWZ
pY2F0aW9uIEF1dGhvcml0eTETMBEGA1UECgwKQXBwbGUgSW5jLjELMAkGA1UEBhMCVVMCCEKQlelCCG%2BKMA0GCW
CGSAFlAwQCAQUAoGkwGAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEHATAcBgkqhkiG9w0BCQUxDxcNMTQxMDAzMjE1NjQ
zWjAvBgkqhkiG9w0BCQQxIgQgg8i4X6yRAU7AXS1lamCf02UIQlpUvNPToXUaamsFUT8wCgYIKoZIzj0EAwIERzBF
AiBe17NGTuuk%2BW901k3Oac4Z90PoMhN1qRqnij9KNEb%2FXAIhALELZyDWw0fQM8t0pXO86gg9xXFz424rEMlJ0
1TM1VxhAAAAAAAA&applepay.version=EC_v1&applepay.header.applicationData=496461ea64b50527d2
d792df7c38f301300085dd463e347453ae72debf6f4d14&applepay.header.ephemeralPublicKey=MFkwEwY
HKoZIzj0CAQYIKoZIzj0DAQcDQgAEarp8xOhLX9QliUPS9c54i3cqEfrJD37NG75ieNxncOeFLkjCk%2FBn3jVxHl
ecRwYqe%2BAWQxZBtDyewaZcmWz5lg%3D%3D&applepay.header.publicKeyHash=zoV5b2%2BmqnMIxU9avTeq
Wxc7OW3fnKXFxyhY0cyRixU%3D&applepay.header.transactionId=23e26bd8741fea9e7a4d78a69f4255b3
15d39ec14233d6f1b32223d1999fb99f" https://request.eprotect.vantivprelive.com/
eProtect/paypage
```

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

2.3.2.3 Sample Apple Pay POST Response

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

```
{"bin":"410000","firstSix":"410000","lastFour":"0001","paypageRegistrationId":"S0ZBUURMTl
ZkMTgrbW1IL3BZVFFmaDh0M0hjdDZ5RXcxQzRQUkJRKzdVc3JURXp0N0JBdmhDN05aTllUQU5rY1RCMDhLNXg2clI
0cDV3Sk5vQmlPTjY3V2plbDVac0lqd0FkblYwVTdQWms9","type":"VI","id":"1234","vantivTxnId":"828
26626153431509","message":"Success","orderId":"PValid","reportGroup":"*merchant1500","res
ponse":"870","responseTime":"2015-01-19T18:35:27","expDate":"0718"}
```

2.3.3 Using the Vantiv Mobile API for Android Pay

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

This is the recommended and typical method of implementing Android Pay for Web and Mobile Applications on the Vantiv eCommerce platform. The steps that follow, along with Figure 2-3, illustrate the high level flow of messages associated with an Android Pay purchase, when utilizing the Vantiv eProtect service.

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

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

Setting the PaymentMethodTokenizationParameters

```
PaymentMethodTokenizationParameters parameters =
PaymentMethodTokenizationParameters .newBuilder()
.setPaymentMethodTokenizationType(PaymentMethodTokenizationType.PAYMENT_GATEWAY)
.addParameter("gateway","vantiv")
.addParameter("vantiv:merchantPayPageId",payPageId)
.addParameter("vantiv:merchantOrderId",orderId)
.addParameter("vantiv:merchantTransactionId",id)
.addParameter("vantiv:merchantReportGroup",reportGroup)
.build();
```

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

Setting New Object in the MaskedWalletRequest

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

```
.setPaymentMethodTokenizationParameters(parameters)
.build();
```

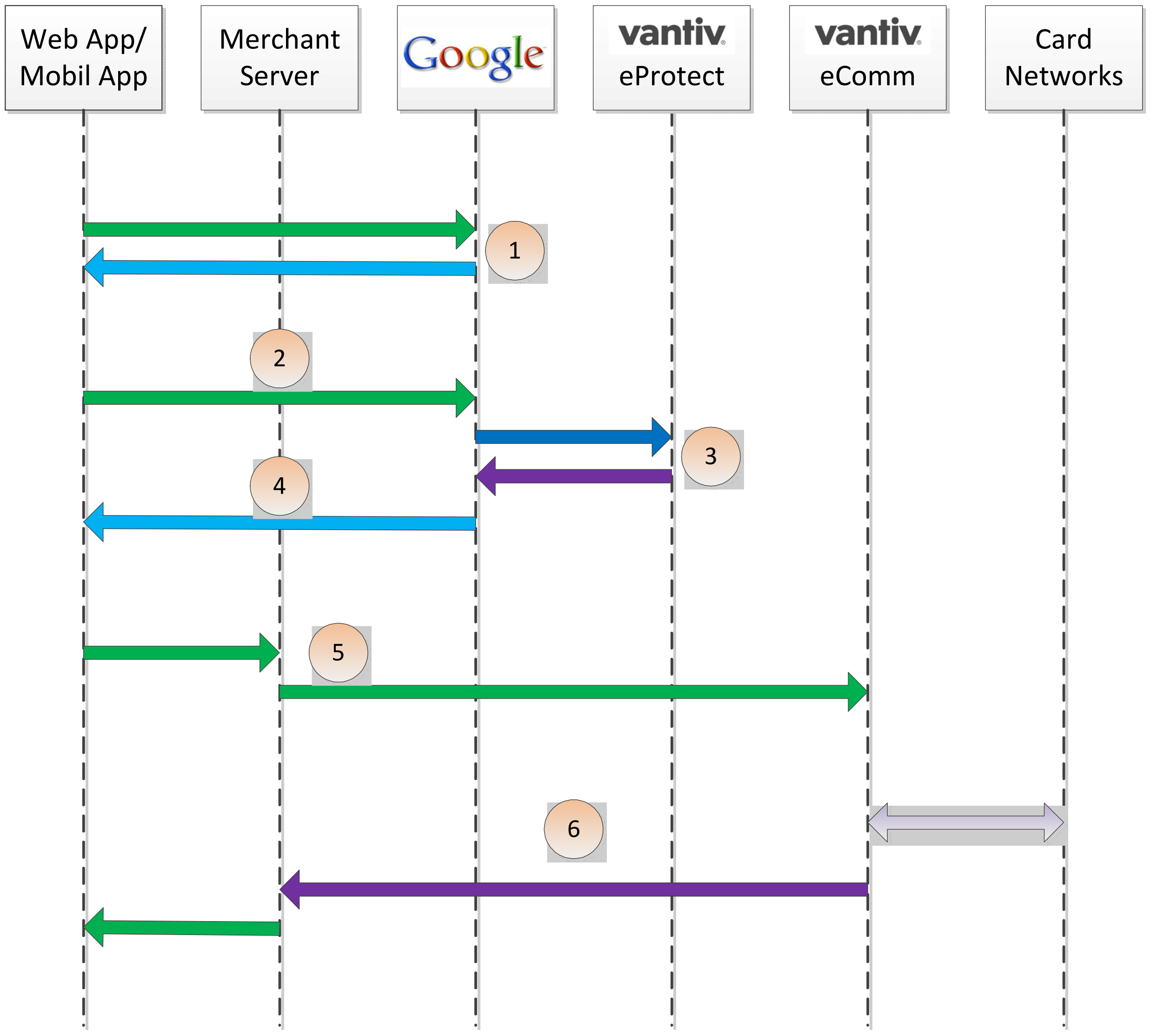
The information returned by Google in the `MaskedWallet` object may include a masked card number (last-four digits exposed) and shipping information. The consumer has the option of changing this information. If any info changes, Android Pay returns an updated `MaskedWallet` object.

2. Upon confirmation of the order by the consumer your application initiates a `FullWalletRequest` to Google.
3. After receiving the `FullWalletRequest` from your application, Google submits the card information to Vantiv eProtect. The eProtect servers return a low-value token (`paypageRegistrationId`).
4. Google returns the low-value token to your application along with the Full Wallet information.
5. Your applications sends the transaction information to your servers along with the low-value token. Your servers submit the Auth/Sale transaction to the Vantiv eCommerce platform. You must set the `orderSource` to `androidpay` in the transaction.

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

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

FIGURE 2-3 High Level Message Flow for Android Pay and Pay with Google™ using eProtect



2.3.4 Using the Vantiv Mobile API for Pay with Google

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

This is the recommended and typical method of implementing Pay with Google for Mobile Applications on the Vantiv eCommerce platform. The steps that follow, along with Figure 2-3, illustrate the high level flow of messages associated with an Pay with Google purchase, when utilizing the Vantiv eProtect™ service.

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

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

Setting the PaymentMethodTokenizationParameters

```
PaymentMethodTokenizationParameters parameters =
PaymentMethodTokenizationParameters .newBuilder()
.setPaymentMethodTokenizationType(PaymentMethodTokenizationType.PAYMENT_GATEWAY)
  .addParameter("gateway","vantiv")
  .addParameter("vantiv:merchantPayPageId",payPageId)
  .addParameter("vantiv:merchantOrderId",orderId)
  .addParameter("vantiv:merchantTransactionId",id)
  .addParameter("vantiv:merchantReportGroup",reportGroup)
  .build();
```

IMPORTANT: **Use the same `orderId` value on all calls (i.e., Google, Register Token, Authorization, Sale, etc.). By using the same `orderId`, customers can track their orders when using a Google-provided app.**

Setting New Object in the PaymentDataRequest

```
PaymentDataRequest request = PaymentDataRequest.newBuilder()
  .addAllowedPaymentMethods (new List,int.(){
      WalletConstants.PAYMENT_METHOD_CARD,
      WalletConstants.PAYMENT_METHOD_TOKENIZED_CARD)
  .setMerchantName(Constants.MERCHANT_NAME)
  .setPhoneNumberRequired(true)
  .setShippingAddressRequired(true)
  .setCurrencyCode(Constants.CURRENCY_CODE_USD)
  .setEstimatedTotalPrice(cartTotal)
    .setCart(Cart.newBuilder()
    .setCurrencyCode(Constants.CURRENCY_CODE_USD)
    .setTotalPrice(cartTotal)
    .setLineItems(lineItems)
    .build())
  .setPaymentMethodTokenizationParameters(parameters)
  .build();
```

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

2. Upon confirmation of the order by the consumer your application initiates a `FullWalletRequest` to Google.
3. After receiving the `FullWalletRequest` from your application, Google submits the card information to Vantiv eProtect. The eProtect servers return a low-value token (`paypageRegistrationId`).
4. Google returns the low-value token to your application along with the Full Wallet information.
5. Your applications sends the transaction information to your servers along with the low-value token. Your servers submit the Auth/Sale transaction to the Vantiv eComm platform. You must set the `orderSource` to **`androidpay`** in the transaction.

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

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

2.4 Collecting Diagnostic Information

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

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

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

2.5 Transaction Examples When Using cnpAPI

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

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

See cnpAPI Elements for eProtect on page 84 for definitions of the eProtect-related elements used in these examples.

This section is meant as a supplement to the *Vantiv cnpAPI Reference Guide*. Refer to the *Vantiv cnpAPI 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 eProtect feature. See the *Vantiv cnpAPI 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 eProtect feature, as well as the Authorization Response format.

NOTE: Although the schema defines the `<expDate>` element as an *optional* child of `<paypage>` element, Vantiv does not store expiration dates. Therefore, you must always submit an expiration date value with each eProtect cnpAPI transaction.

2.5.2.1 Authorization Request Structure

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

```
<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

```
<cnpOnlineRequest version="12.0" xmlns="http://www.vantivcnp.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>
  </cnpOnlineRequest>
```

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">
  <cnpTxnId>Transaction Id</cnpTxnId>
  <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).**

```
<cnpOnlineResponse version="12.0" xmlns="http://www.vantivcnp.com/schema"
  response="0" message="Valid Format">
  <authorizationResponse id="834262" reportGroup="ABC Division"
  customerId="038945">
    <cnpTxnId>969506</cnpTxnId>
    <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>
      <cnpToken>1111000100090005</cnpToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </authorizationResponse>
</cnpOnlineResponse>
```

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.

NOTE: **Although the schema defines the `<expDate>` element as an *optional* child of `<paypage>` element, Vantiv does not store expiration dates. Therefore, you must always submit an expiration date value with each eProtect cnpAPI transaction.**

2.5.3.1 Sale Request Structure

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

```
<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

```
<cnpOnlineRequest version="12.0" xmlns="http://www.vantivcnp.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>
</cnpOnlineRequest>
```

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">
  <cnpTxnId>Transaction Id</cnpTxnId>
  <response>Response Code</response>
  <orderId>Order Id</orderId>
  <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).

```
<cnpOnlineResponse version="12.0" xmlns="http://www.vantivcnp.com/schema"
  response="0" message="Valid Format">
  <saleResponse id="834262" reportGroup="ABC Division" customerId="038945">
    <cnpTxnId>969506</cnpTxnId>
    <response>000</response>
    <orderId>65347567</orderId>
    <responseTime>2017-07-25T15:13:43</responseTime>
    <postDate>2017-07-25</postDate>
    <message>Approved</message>
    <authCode>123457</authCode>
    <fraudResult>
      <avsResult>11</avsResult>
      <cardValidationResult>P</cardValidationResult>
    </fraudResult>
    <tokenResponse>
      <cnpToken>1111000100090005</cnpToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </saleResponse>
</cnpOnlineResponse>
```

2.5.4 Register Token Transactions

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

2.5.4.1 Register Token Request

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

When you submit the CVV2/CVC2/CID in a `registerTokenRequest`, our platform encrypts and stores the value on a temporary basis (24 hours) 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 Authorization/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 Register Token request structures, see the *Vantiv cnpAPI Reference Guide*.

Example: Online Register Token Request - eProtect

```
<cnpOnlineRequest version="12.0" xmlns="http://www.vantivcnp.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>
    </cnpOnlineRequest>
```

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, 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">
    <cnpTxnId>Transaction ID</cnpTxnId>
    <cnpToken>Token</cnpToken>
    <bin>BIN</bin>
    <type>Method of Payment</type>
    <response>Response Code</response>
    <responseTime>Response Time</responseTime>
    <message>Response Message</message>
  </registerTokenResponse>
```

Example: Online Register Token Response - PayPage

```
<cnpOnlineResponse version="12.0" xmlns="http://www.vantivcnp.com/schema"
  id="123" response="0" message="Valid Format" cnpSessionId="987654321">
    <registerTokenResponse id="99999" reportGroup="RG1">
      <cnpTxnId>21122700</cnpTxnId>
      <cnpToken>1111000100360002</cnpToken>
      <bin>400510</bin>
      <type>VI</type>
      <response>801</response>
      <responseTime>2010-10-26T17:21:51</responseTime>
      <message>Account number was successfully registered</message>
    </registerTokenResponse>
</cnpOnlineResponse>
```

2.5.5 Force Capture Transactions

A Force Capture transaction is a Capture transaction used when you do not have a valid Authorization for the order, but have fulfilled the order and wish to transfer funds. You can use a `<paypageRegistrationID>` with a Force Capture transaction.

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

NOTE: Although the schema defines the `<expDate>` element as an *optional* child of `<paypage>` element, Vantiv does not store expiration dates. Therefore, you must always submit an expiration date value with each eProtect cnpAPI transaction.

2.5.5.1 Force Capture Request

You must structure a Force Capture request as shown in the following examples when using eProtect. 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

```
<cnpOnlineRequest version="12.0" xmlns="http://www.vantivcnp.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>
  </cnpOnlineRequest>
```

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">
  <cnpTxnId>Transaction Id</cnpTxnId>
  <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

```
<cnpOnlineResponse version="12.0" xmlns="http://www.vantivcnp.com/schema"
```

```
    response="0" message="Valid Format">
    <forceCaptureResponse id="2" reportGroup="ABC Division"
     customerId="038945">
      <cnpTxnId>1100030204</cnpTxnId>
      <response>000</response>
      <responseTime>2009-07-11T14:48:48</responseTime>
      <postDate>2009-07-11</postDate>
      <message>Approved</message>
      <tokenResponse>
        <cnpToken>1111000100090005</cnpToken>
        <tokenResponseCode>801</tokenResponseCode>
        <tokenMessage>Account number was successfully registered</tokenMessage>
        <type>VI</type>
        <bin>402410</bin>
      </tokenResponse>
    </forceCaptureResponse>
  </cnpOnlineResponse>
```

2.5.6 Capture Given Auth Transactions

You can use a Capture Given Auth transaction with a `<paypageRegistrationID>` if the `<cnpTxnId>` 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).

NOTE: Although the schema defines the `<expDate>` element as an *optional* child of `<paypage>` element, Vantiv does not store expiration dates. Therefore, you must always submit an expiration date value with each eProtect cnpAPI transaction.

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

```
<cnpOnlineRequest version="12.0" xmlns="http://www.vantivcnp.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>2017-06-22</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>
</cnpOnlineRequest>
```

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">

  <cnpTxnId>Transaction Id</cnpTxnId>

  <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

```
<cnpOnlineResponse version="12.0" xmlns="http://www.vantivcnp.com/schema"
  response="0" message="Valid Format">
  <captureGivenAuthResponse id="2" reportGroup="ABC Division"
   customerId="038945">
    <cnpTxnId>1100030204</cnpTxnId>
    <response>000</response>
    <responseTime>2011-07-11T14:48:48</responseTime>
    <postDate>2011-07-11</postDate>
    <message>Approved</message>
    <tokenResponse>
      <cnpToken>1111000100090005</cnpToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </captureGivenAuthResponse>
</cnpOnlineResponse>
```

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

NOTE: **Although the schema defines the `<expDate>` element as an *optional* child of `<paypage>` element, Vantiv does not store expiration dates. Therefore, you must always submit an expiration date value with each eProtect cnpAPI transaction.**

2.5.7.1 Credit Request Transaction

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

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

Example: Online Credit Request Transaction

```
<cnpOnlineRequest version="12.0" xmlns="http://www.vantivcnp.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>
</cnpOnlineRequest>
```

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">
  <cnpTxnId>Transaction Id</cnpTxnId>
  <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

```
<cnpOnlineResponse version="12.0" xmlns="http://www.vantivcnp.com/schema"
  response="0" message="Valid Format">
  <creditResponse customerId="038945" id="5" reportGroup="ABC Division">
```

```
      <cnpTxnId>1100030204</cnpTxnId>
      <response>001</response>
      <responseTime>2009-08-11T14:48:48</responseTime>
      <postDate>2009-08-11</postDate>
      <message>Transaction received</message>
      <tokenResponse>
        <cnpToken>1111000100090005</cnpToken>
        <tokenResponseCode>801</tokenResponseCode>
        <tokenMessage>Account number was successfully registered</tokenMessage>
        <type>VI</type>
        <bin>402410</bin>
      </tokenResponse>
    </creditResponse>
  </cnpOnlineResponse>
```

2.6 Testing and Certification

Vantiv requires successful certification testing for the eProtect transactions before you can use them in production. During certification testing, you will work through each required test scenario with an Implementation Consultant. This section provides the specific data you must use in your eProtect 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 eProtect includes browser and/or mobile native application interaction, JavaScript interaction, and transaction requests as well as cnpAPI responses with the Registration ID.

IMPORTANT: **Because browsers differ in their handling of eProtect transactions, Vantiv recommends testing eProtect on various devices (including smart phones and tablets) and all browsers, including Internet Explorer/Edge, Google Chrome, Apple Safari, and Mozilla Firefox.**

See Certification and Testing Environments on page 10 for information, maintenance windows, and limitations for the pre-live testing environment.

The eProtect Certification tests the following:

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

- A successful transaction
- cnpAPI transaction requests and responses

For browser-based checkout pages only:

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

See the section, eProtect-Specific Response Codes on page 12 for definitions of the response codes.

2.6.1 Testing eProtect Transactions

To test eProtect transactions:

1. Verify that your checkout page or mobile application is coded correctly. See one of the following sections for more information:
   - Integrating Customer Browser JavaScript API Into Your Checkout Page on page 22.
   - Integrating iFrame into your Checkout Page on page 32.
   - Integrating eProtect Into Your Mobile Application on page 39.

2. Verify that you are using the appropriate URL (see Table 1-2, "eProtect Certification, Testing, and Production URLs" on page 11) for the testing and certification environment, for example:

```
https://request.eprotect.vantivprelive.com/eProtect/eProtect-api2.js
```

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

3. Submit transactions from your checkout page or mobile application using the **Card Numbers** and **Card Validation Numbers** from Table 2-8. When performing these tests, you can use any expiration date and card type.
4. Verify that your results match the **Result** column in Table 2-8.

TABLE 2-8 Expected eProtect Test Results

Test Case	Card Number	Card Validation Number	Response Code	Result
NOTE: Card Numbers are split into two parts; join **Part 1** and **Part 2** to obtain actual number to use.				
1	Part 1: 51120100 Part 2: 00000003	Any 3-digit	870 (Success)	Registration ID is generated and the card is scrubbed before the form is submitted.
2	Part 1: 445701000 Part 2: 00000009	Any 3-digit	871	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (not Mod10).”
3	Part 1: 44570100000 Part 2: 0000000006	Any 3-digit	873	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (too long).” ***Note***: Do not use when testing iFrame.
4	Part 1: 601101 Part 2: 000003	Any 3-digit	872	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (too short).”
5	Part 1: 44570100 Part 2: B00000006	Any 3-digit	874	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (not a number).”

TABLE 2-8 Expected eProtect Test Results (Continued)

Test Case	Card Number	Card Validation Number	Response Code	Result
6	Part 1: 60110100 Part 2: 00000003	Any 3-digit	875	Checkout form displays error message to card holder, for example, “We are experiencing technical difficulties. Please try again later or call 555-555-1212.”
7	Part 1: 51234567 Part 2: 898010003	Any 3-digit	876	Checkout form displays error message to card holder, for example, “Invalid Card Number - Check and retry (failure from server).”
8	Part 1: 3750010 Part 2: 00000005	Any 3-digit	None (Timeout error)	Checkout form displays error message to card holder, for example, “We are experiencing technical difficulties. Please try again later or call 555-555-1212 (timeout).”
9	Part 1: 44570102 Part 2: 00000007	Any 3-digit	889	Checkout form displays error message to card holder, for example, “We are experiencing technical difficulties. Please try again later or call 555-555-1212.”
10	Part 1: 51120100 Part 2: 00000003	abc	881	Checkout form displays error message to card holder, for example “Invalid Card Validation Number - Check and retry (not a number)”.
11	Part 1: 51120100 Part 2: 00000003	12	882	Checkout form displays error message to card holder, for example “Invalid Card Validation Number - Check and retry (too short)”.
12	Part 1: 51120100 Part 2: 00000003	12345	883	Checkout form displays error message to card holder, for example “Invalid Card Validation Number - Check and retry (too long)”. ***Note***: Do not use when testing iFrame.

To test the submission of eProtect data using cnpAPI Authorization transactions:

1. Verify that your cnpAPI template is coded correctly for this transaction type (see Authorization Transactions on page 53).
2. Submit three Authorization transactions using the eProtect data from Table 2-9.
3. Verify that your `authorizationResponse` values match the **Response Code** column.

NOTE: **If you are using OMNI tokens, the eCommerce platform can only determine that the card can not be found and will not be able to determine the card type. This may return a response of 822 -*Token not found* or 330 - *Invalid Payment Type*.**

TABLE 2-9 eProtect Authorization Transaction Request and Response Data

Test Case	PayPage Registration ID	Exp. Date	Card Validation Number	Response Code Expected
13	(Use the PayPage Registration ID value received when executing Test Case #1)	Any 4 digits	Any 3 digits	000 - Approved
14	pDZJcmd1VjNlYXNaSlRMTGpocVZQY1NWVXE4ZW5UTko4NU9KK3p1L1p1Vzg4YzVPQVlSUHNlTG1JN2I0NzlyTg==	1230	Any 3 digits	877 - Invalid PayPage Registration ID
15	RGFQNCt6U1d1M21SeVByVTM4dHlHb1FsVkUrSmpnWXhNY0o5UkMzRlZFanZiUHVnYjN1enJXbG1WSDF4aXlNcA==	1230	Any 3 digits	878 - Expired PayPage Registration ID

A

CODE SAMPLES AND OTHER INFORMATION

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

- HTML Checkout Page Examples
- Information Sent to Order Processing Systems
- cnpAPI Elements for eProtect

NOTE: **The PayPage product is now known as *Vantiv eProtect*. The term 'PayPage' however, is still used in this guide in certain text descriptions, along with many data elements, JS code, and URLs. Use of these data elements, etc., with the PayPage name is still valid with this release, but will transition to 'eProtect' in a future release.**

A.1 HTML Checkout Page Examples

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

This section provides three HTML checkout page examples:

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

A.1.1 HTML Example for Non-eProtect Checkout Page

For comparison purposes, the following HTML sample is for a simple check-out page that is not integrated with eProtect. 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 JavaScript API-Integrated Checkout Page

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

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

```
    <HTML>
         <head>
         <title>PayPage Merchant Simple Checkout</title>
         <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
         type="text/javascript"></script>
         <script
         src="https://request.eprotect.vantivprelive.com/eProtect/eProtect-api2.js"
         type="text/javascript"></script>
         <script
         $(document).ready(
          function(){
           function setEprotectResponseFields(response) {
             document.getElementById('response$code').value = response.response;
             document.getElementById('response$message').value = response.message;
             document.getElementById('response$responseTime').value =
         response.responseTime;
             document.getElementById('response$vantivTxnId').value =
         response.vantivTxnId;
             document.getElementById('response$type').value = response.type;
             document.getElementById('response$firstSix').value = response.firstSix;
             document.getElementById('response$lastFour').value = response.lastFour;

        }
        function submitAfterEprotect (response) {
          setEprotectResponseFields(response);
          document.forms['fCheckout'].submit();
        }
        function timeoutOnEprotect () {
          alert("We are experiencing technical difficulties.  Please try again later or
call 555-555-1212 (timeout)");
        }

           function onErrorAfterEprotect (response) {
             setEprotectResponseFields(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)");
             }
```

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

```
                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;
          }
          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
              setEprotectResponseFields({"response":"", "message":""});

              var eProtectRequest = {
                "paypageId" : document.getElementById("request$paypageId").value,
                "reportGroup" : document.getElementById("request$reportGroup").value,
                "orderId" : document.getElementById("request$orderId").value,
                "id" : document.getElementById("request$merchantTxnId").value
                "applepay" : applepay
                "url" : "https://request.eprotect.vantivprelive.com"
              };
              new eProtect().sendToEprotect(eProtectRequest, formFields,
submitAfterEprotect, onErrorAfterEprotect, timeoutOnEprotect, 15000);
                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"/>
```

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

```
    <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="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="callEprotect()">Check
out with eProtect</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$vantivTxnId" name="response$vantivTxnId"
readOnly="true"/>
    <input type="hidden" id="response$firstSix" name="response$firstSix"
readOnly="true"/>
    <input type="hidden" id="response$lastFour" name="response$lastFour"
readOnly="true"/>
  </form>
</BODY>
<script>

/* This is an example of how to handle being unable to download the eprotect-api2 */
  function callEprotect() {
      if(typeof new eProtect() != 'object') {
      alert("We are experiencing technical difficulties.  Please try again later or call
555-555-1212 (API unavailable)" );
  }

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

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

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

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

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

    <script src=https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js
type="text/javascript"></script>
    <script src="https://request.eprotect.vantivprelive.com/eProtect/js/
eProtect-iframe-client.min.js"></script>
</head>
<body>

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

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

```
        </table>
      </form>
    </div>
      <br/>
    <h3>Test Input Fields</h3>
    <table class="testFieldTable">
        <tr>
          <td>Paypage ID</td><td><input type="text" id="request$paypageId"
name="request$paypageId" value="a2y4o6m8k0" disabled/></td>
          <td>Merchant Txn ID</td><td><input type="text" id="request$merchantTxnId"
name="request$merchantTxnId" value="987012"/></td>
        </tr>
        <tr>
          <td>Order ID</td><td><input type="text" id="request$orderId"
name="request$orderId" value="order_123"/></td>
          <td>Report Group</td><td><input type="text" id="request$reportGroup"
name="request$reportGroup" value="*merchant1500" disabled/></td>
        </tr>
        <tr>
          <td>JS Timeout</td><td><input type="text" id="request$timeout"
name="request$timeout" value="15000" disabled/></td>
        </tr>
      </table>
<h3>Test Output Fields</h3>
<table class="testFieldTable">
    <tr>
        <td>Response Code</td><td><input type="text" id="response$code"
name="response$code" readOnly="true"/></td>
        <td>ResponseTime</td><td><input type="text" id="response$responseTime"
name="response$responseTime" readOnly="true"/></td>
    </tr>
    <tr>
        <td>Response Message</td><td colspan="3"><input type="text"
id="response$message" name="response$message" readOnly="true" size="100"/></td>
    </tr>
    <tr><td>&nbsp;</td><td></tr>
    <tr>
        <td>Vantiv Txn ID</td><td><input type="text" id="response$vantivTxnId"
name="response$vantivTxnId" readOnly="true"/></td>
        <td>Merchant Txn ID</td><td><input type="text" id="response$merchantTxnId"
name="response$merchantTxnId" readOnly="true"/></td>
    </tr>
    <tr>
        <td>Order ID</td><td><input type="text" id="response$orderId"
name="response$orderId" readOnly="true"/></td>
        <td>Report Group</td><td><input type="text" id="response$reportGroup"
name="response$reportGroup" readOnly="true"/></td>
    </tr>
    <tr><td>Type</td><td><input type="text" id="response$type" name="response$type"
readOnly="true"/></td></tr>
    <tr>
        <td>Expiration Month</td><td><input type="text" id="response$expMonth"
name="response$expMonth" readOnly="true"/></td>
        <td>Expiration Year</td><td><input type="text" id="response$expYear"
name="response$expYear" readOnly="true"/></td>
    </tr>
    <tr><td>&nbsp;</td><td></tr>
    <tr>
        <td>First Six</td><td><input type="text"
id="response$firstSix"name="response$firstSix" readOnly="true"/></td>
      <td>Last Four</td><td><input type="text"
id="response$lastFour"name="response$lastFour" readOnly="true"/></td>
```

```
      </tr>
      <tr><td>Timeout Message</td><td><input type="text" id="timeoutMessage"
name="timeoutMessage" readOnly="true"/></td></tr>
      <tr><td>Expected Results</td>
          <td colspan="3">
          <textarea id="expectedResults" name="expectedResults" rows="5" cols="100"
readOnly="true">
            CC Num           - Token Generated (with simulator)
            410000&#48;00000001 - 1111222&#50;33330001
            5123456&#55;89012007 - 1112333&#51;44442007
            3783102&#48;3312332 - 1113444&#53;552332
            601100&#48;990190005 - 1114555&#53;66660005
          </textarea></td>
      </tr>
      <tr>
            <td>Encrypted Card</td>
            <td colspan="3"><textarea id="base64enc" name="base64enc" rows="5"
cols="100" readOnly="true"></textarea></td>
      </tr>
  </table>
<script>


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

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

```
                "1":"January",
                "2":"February",
                "3":"March",
                "4":"April",
                "5":"May",
                "6":"June",
                "7":"July",
                "8":"August",
                "9":"September",
                "10":"October",
                "11":"November",
                "12":"December"
            },
            "numYears": 8,
            "tooltipText": "A CVV is the 3 digit code on the back of your Visa, MasterCard
and Discover or a 4 digit code on the front of your American Express",
            "tabIndex": {
                "cvv":1,
                "accountNumber":2,
                "expMonth":3,
                "expYear":4
            },
            "placeholderText": {
                  "cvv":"CVV",
                  "accountNumber":"Account Number"
            },
              "inputsEmptyCallback": inputsEmptyCallback,
              "enhancedUxFeatures" : {
              "inlineFieldValidations": true,
              }

    };
    if(typeof eProtectiframeClient === 'undefined') {
     //This means we couldn't download the eprotect-iframe-client javascript library
                alert("Couldn't download eprotect-iframe-client.min javascript");
    }
    var eProtectiframeClient = new EprotectIframeClient(configure);
    eProtectiframeClient.autoAdjustHeight();
    document.getElementById("fCheckout").onsubmit = function(){
        var message = {
              "id":document.getElementById("request$merchantTxnId").value,
              "orderId":document.getElementById("request$orderId").value
        };
        startTime = new Date().getTime();
        eProtectiframeClient.getPaypageRegistrationId(message);
        return false;
    };
});

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

A.2 Information Sent to Order Processing Systems

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

A.2.1 Information Sent Without Integrating eProtect

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

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

A.2.2 Information Sent with Browser-Based eProtect Integration

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

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

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

response$firstSix - 410000
response$lastFour - 0001

A.2.3 Information Sent with Mobile API-Based Integration

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

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

A.3 cnpAPI Elements for eProtect

This section provides definitions for the elements used in the cnpAPI for eProtect transactions.

Use this information in combination with the various cnpAPI schema files to assist you as you build the code necessary to submit eProtect transactions to our transaction processing systems. Each section defines a particular element, its relationship to other elements (parents and children), as well as any attributes associated with the element.

For additional information on the structure of cnpAPI requests and responses using these elements, along with examples, see Transaction Examples When Using cnpAPI on page 52. For a comprehensive list of all cnpAPI elements and usage, see Chapter 4, "cnpAPI Elements" in the *Vantiv cnpAPI Reference Guide*.

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

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

A.3.1 cardValidationNum

The `<cardValidationNum>` element is an optional child of the `<card>`, `<paypage>`, `<token>`, `<registerTokenRequest>`, or `<updateCardValidatinNumOnToken>` element, which you use to submit either the CVV2 (Visa), CVC2 (MasterCard), or CID (American Express and Discover) value.

NOTE: **Some American Express cards may have a 4-digit CID on the front of the card and/or a 3-digit CID on the back of the card. You can use either of the numbers for card validation, but not both.**

When you submit the CVV2/CVC2/CID in a `registerTokenRequest`, our platform encrypts and stores the value on a temporary basis (24 hours) for later use in a tokenized Authorization or Sale transaction submitted without the value. This is done to accommodate merchant systems/workflows where the security code is available at the time of token registration, but not at the time of the Auth/Sale. If for some reason you need to change the value of the security code supplied at the time of the token registration, use an `<updateCardValidationNumOnToken>` transaction. To use the stored value when submitting an Auth/Sale transaction, set the `<cardValidationNum>` value to 000.

The `cardValidationNum` element is an optional child of the `virtualGiftCardResponse` element, where it defines the value of the validation Number associated with the Virtual Gift Card requested

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

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

Parent Elements:

card, paypage, token, registerTokenRequest, updateCardValidationNumOnToken, virtualGiftCardResponse

Attributes:

None

Child Elements:

None

A.3.2 expDate

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

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

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

Parent Elements:

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

Attributes:

None

Child Elements:

None

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

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

A.3.3 paypage

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

Parent Elements:

authorization, sale, captureGivenAuth, forceCapture, credit, updateSubscription

Attributes:

None

Child Elements:

Required: paypageRegistrationId

Optional: cardValidationNum, expDate, type

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

Example: paypage Structure

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

A.3.4 paypageRegistrationId

The `<paypageRegistrationId>` element is a required child of the `<paypage>` element. It specifies the Registration ID generated by eProtect. It can also be used in a Register Token Request to obtain a token based on eProtect activity prior to submitting an Authorization or Sale transaction.

Type = String; **minLength** = N/A; **maxLength** = 512

Parent Elements:

paypage, registerTokenRequest

Attributes:

None

Child Elements:

None

A.3.5 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 or 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:

cnpOnlineRequest, 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 eCommerce iQ where this transaction displays. **minLength** = 1 **maxLength** = 25

Child Elements:

Required: either accountNumber, mpos, echeckForToken, paypageRegistrationId, or applepay

Optional: orderId, cardValidationNum

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

A.3.6 registerTokenResponse

The `<registerTokenResponse>` element is the parent element for the response to `<registerTokenRequest>` transactions. You receive this transaction type in response to the submission of an account number or registration ID for tokenization in a Register Token transaction.

Parent Elements:

cnpOnlineResponse, 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: cnpTxnId, response, message, responseTime

Optional: eCheckAccountSuffix, cnpToken, bin, type, applepayResponse, androidpayResponse

B

CSS PROPERTIES FOR IFRAME API

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

See the section Creating a Customized CSS for iFrame on page 14 before using the properties listed here.

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

NOTE: **If you are evaluating your styling options and/or having trouble creating your own style sheet, Vantiv can provide sample CSS files. Please contact your assigned Implementation Consultant for sample CSS files.**

B.1 CSS Property Groups

For additional detail on each property type, click the desired link below to navigate to the corresponding section:

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

TABLE B-1 Color Properties

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

TABLE B-2 Background and Border Properties

Property	Description
background **(Do not use)**	*Sets all the background properties in one declaration*
background-attachment **(Do not use)**	*Sets whether a background image is fixed or scrolls with the rest of the page*
background-color	Sets the background color of an element
background-image	Sets the background image for an element

TABLE B-2 Background and Border Properties (Continued)

Property	**Description**
background-position **(Do not use)**	*Sets the starting position of a background image*
background-repeat **(Do not use)**	*Sets how a background image will be repeated*
background-clip	Specifies the painting area of the background
background-origin **(Do not use)**	*Specifies the positioning area of the background images*
background-size **(Do not use)**	*Specifies the size of the background images*
border	Sets all the border properties in one declaration
border-bottom	Sets all the bottom border properties in one declaration
border-bottom-color	Sets the color of the bottom border
border-bottom-left-radius	Defines the shape of the border of the bottom-left corner
border-bottom-right-radius	Defines the shape of the border of the bottom-right corner
border-bottom-style	Sets the style of the bottom border
border-bottom-width	Sets the width of the bottom border
border-color	Sets the color of the four borders
border-image **(Do not use)**	*A shorthand property for setting all the border-image-* properties*
border-image-outset **(Do not use)**	*Specifies the amount by which the border image area extends beyond the border box*
border-image-repeat **(Do not use)**	*Specifies whether the image-border should be repeated, rounded or stretched*
border-image-slice	Specifies the inward offsets of the image-border
border-image-source **(Do not use)**	*Specifies an image to be used as a border*
border-image-width **(Do not use)**	*Specifies the widths of the image-border*
border-left	Sets all the left border properties in one declaration
border-left-color	Sets the color of the left border

TABLE B-2 Background and Border Properties (Continued)

Property	Description
border-left-style	Sets the style of the left border
border-left-width	Sets the width of the left border
border-radius	A shorthand property for setting all the four border-*-radius properties
border-right	Sets all the right border properties in one declaration
border-right-color	Sets the color of the right border
border-right-style	Sets the style of the right border
border-right-width	Sets the width of the right border
border-style	Sets the style of the four borders
border-top	Sets all the top border properties in one declaration
border-top-color	Sets the color of the top border
border-top-left-radius	Defines the shape of the border of the top-left corner
border-top-right-radius	Defines the shape of the border of the top-right corner
border-top-style	Sets the style of the top border
border-top-width	Sets the width of the top border
border-width	Sets the width of the four borders
box-decoration-break	Sets the behavior of the background and border of an element at page-break, or, for in-line elements, at line-break.
box-shadow	Attaches one or more drop-shadows to the box

TABLE B-3 Basic Box Properties

Property	Description
bottom	Specifies the bottom position of a positioned element
clear	Specifies which sides of an element where other floating elements are not allowed
clip	Clips an absolutely positioned element
display	Specifies how a certain HTML element should be displayed
float	Specifies whether or not a box should float
height	Sets the height of an element
left	Specifies the left position of a positioned element

TABLE B-3 Basic Box Properties (Continued)

Property	Description
overflow	Specifies what happens if content overflows an element's box
overflow-x	Specifies whether or not to clip the left/right edges of the content, if it overflows the element's content area
overflow-y	Specifies whether or not to clip the top/bottom edges of the content, if it overflows the element's content area
padding	Sets all the padding properties in one declaration
padding-bottom	Sets the bottom padding of an element
padding-left	Sets the left padding of an element
padding-right	Sets the right padding of an element
padding-top	Sets the top padding of an element
position	Specifies the type of positioning method used for an element (static, relative, absolute or fixed)
right	Specifies the right position of a positioned element
top	Specifies the top position of a positioned element
visibility	Specifies whether or not an element is visible
width	Sets the width of an element
vertical-align	Sets the vertical alignment of an element
z-index	Sets the stack order of a positioned element

TABLE B-4 Flexible Box Layout

Property	Description
align-content	Specifies the alignment between the lines inside a flexible container when the items do not use all available space.
align-items	Specifies the alignment for items inside a flexible container.
align-self	Specifies the alignment for selected items inside a flexible container.
display	Specifies how a certain HTML element should be displayed
flex	Specifies the length of the item, relative to the rest
flex-basis	Specifies the initial length of a flexible item
flex-direction	Specifies the direction of the flexible items

TABLE B-4 Flexible Box Layout (Continued)

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

TABLE B-5 Text Properties

Property	**Description**
hanging-punctuation	Specifies whether a punctuation character may be placed outside the line box
hyphens	Sets how to split words to improve the layout of paragraphs
letter-spacing	Increases or decreases the space between characters in a text
line-break	Specifies how/if to break lines
line-height	Sets the line height
overflow-wrap	Specifies whether or not the browser may break lines within words in order to prevent overflow (when a string is too long to fit its containing box)

TABLE B-5 Text Properties (Continued)

Property	**Description**
tab-size	Specifies the length of the tab-character
text-align	Specifies the horizontal alignment of text
text-align-last	Describes how the last line of a block or a line right before a forced line break is aligned when text-align is “justify”
text-combine-upright	Specifies the combination of multiple characters into the space of a single character
text-indent	Specifies the indentation of the first line in a text-block
text-justify	Specifies the justification method used when text-align is “justify”
text-transform	Controls the capitalization of text
white-space	Specifies how white-space inside an element is handled
word-break	Specifies line breaking rules for non-CJK scripts
word-spacing	Increases or decreases the space between words in a text
word-wrap	Allows long, unbreakable words to be broken and wrap to the next line

TABLE B-6 Text Decoration Properties

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

TABLE B-7 Font Properties

Property	**Description**
@font-face **(Do not use)**	*A rule that allows websites to download and use fonts other than the “web-safe” fonts*

TABLE B-7 Font Properties (Continued)

Property	Description
@font-feature-values	Allows authors to use a common name in font-variant-alternate for feature activated differently in OpenType
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-feature-settings	Allows control over advanced typographic features in OpenType fonts
font-kerning	Controls the usage of the kerning information (how letters are spaced)
font-language-override	Controls the usage of language-specific glyphs in a typeface
font-size	Specifies the font size of text
font-size-adjust	Preserves the readability of text when font fallback occurs
font-stretch	Selects a normal, condensed, or expanded face from a font family
font-style	Specifies the font style for text
font-synthesis	Controls which missing typefaces (bold or italic) may be synthesized by the browser
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-variant-alternates	Controls the usage of alternate glyphs associated to alternative names defined in @font-feature-values
font-variant-caps	Controls the usage of alternate glyphs for capital letters
font-variant-east-asian	Controls the usage of alternate glyphs for East Asian scripts (e.g Japanese and Chinese)
font-variant-ligatures	Controls which ligatures and contextual forms are used in textual content of the elements it applies to
font-variant-numeric	Controls the usage of alternate glyphs for numbers, fractions, and ordinal markers
font-variant-position	Controls the usage of alternate glyphs of smaller size positioned as superscript or subscript regarding the baseline of the font
font-weight	Specifies the weight of a font

TABLE B-8 Writing Modes Properties

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

TABLE B-9 Table Properties

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

TABLE B-10 Lists and Counters Properties

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

TABLE B-11 Animation Properties

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

TABLE B-12 Transform Properties

Property	Description
backface-visibility	Defines whether or not an element should be visible when not facing the screen
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space

TABLE B-13 Transitions Properties

Property	Description
transition	A shorthand property for setting the four transition properties
transition-property	Specifies the name of the CSS property the transition effect is for

TABLE B-13 Transitions Properties

Property	Description
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-timing-function	Specifies the speed curve of the transition effect
transition-delay	Specifies when the transition effect will start

TABLE B-14 Basic User Interface Properties

Property	Description
box-sizing	Tells the browser what the sizing properties (width and height) should include
content	Used with the :before and :after pseudo-elements, to insert generated content
cursor **(Do not use)**	*Specifies the type of cursor to be displayed*
icon **(Do not use)**	*Provides the author the ability to style an element with an iconic equivalent*
ime-mode	Controls the state of the input method editor for text fields
nav-down	Specifies where to navigate when using the arrow-down navigation key
nav-index	Specifies the tabbing order for an element
nav-left	Specifies where to navigate when using the arrow-left navigation key
nav-right	Specifies where to navigate when using the arrow-right navigation key
nav-up	Specifies where to navigate when using the arrow-up navigation key
outline	Sets all the outline properties in one declaration
outline-color	Sets the color of an outline
outline-offset	Offsets an outline, and draws it beyond the border edge
outline-style	Sets the style of an outline
outline-width	Sets the width of an outline
resize	Specifies whether or not an element is resizable by the user

TABLE B-14 Basic User Interface Properties (Continued)

Property	Description
text-overflow	Specifies what should happen when text overflows the containing element

TABLE B-15 Multi-Column Layout Properties

Property	Description
break-after	Specifies the page-, column-, or region-break behavior after the generated box
break-before	Specifies the page-, column-, or region-break behavior before the generated box
break-inside	Specifies the page-, column-, or region-break behavior inside the generated box
column-count	Specifies the number of columns an element should be divided into
column-fill	Specifies how to fill columns
column-gap	Specifies the gap between the columns
column-rule	A shorthand property for setting all the column-rule-* properties
column-rule-color	Specifies the color of the rule between columns
column-rule-style	Specifies the style of the rule between columns
column-rule-width	Specifies the width of the rule between columns
column-span	Specifies how many columns an element should span across
column-width	Specifies the width of the columns
columns	A shorthand property for setting column-width and column-count
widows	Sets the minimum number of lines that must be left at the top of a page when a page break occurs inside an element

TABLE B-16 Paged Media

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

TABLE B-16 Paged Media

Property	Description
page-break-after	Sets the page-breaking behavior after an element
page-break-before	Sets the page-breaking behavior before an element
page-break-inside	Sets the page-breaking behavior inside an element

TABLE B-17 Generated Content for Paged Media

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

TABLE B-18 Filter Effects Properties

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

TABLE B-19 Image Values and Replaced Content

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

TABLE B-20 Masking Properties

Property	Description
mask	
mask-type	

TABLE B-21 Speech Properties

Property	Description
mark	A shorthand property for setting the mark-before and mark-after properties
mark-after	Allows named markers to be attached to the audio stream
mark-before	Allows named markers to be attached to the audio stream
phonemes	Specifies a phonetic pronunciation for the text contained by the corresponding element
rest	A shorthand property for setting the rest-before and rest-after properties
rest-after	Specifies a rest or prosodic boundary to be observed after speaking an element's content
rest-before	Specifies a rest or prosodic boundary to be observed before speaking an element's content
voice-balance	Specifies the balance between left and right channels
voice-duration	Specifies how long it should take to render the selected element's content
voice-pitch	Specifies the average pitch (a frequency) of the speaking voice
voice-pitch-range	Specifies variation in average pitch
voice-rate	Controls the speaking rate
voice-stress	Indicates the strength of emphasis to be applied
voice-volume	Refers to the amplitude of the waveform output by the speech synthesises

TABLE B-22 Marquee Properties

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

B.2 Properties Excluded From White List

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

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

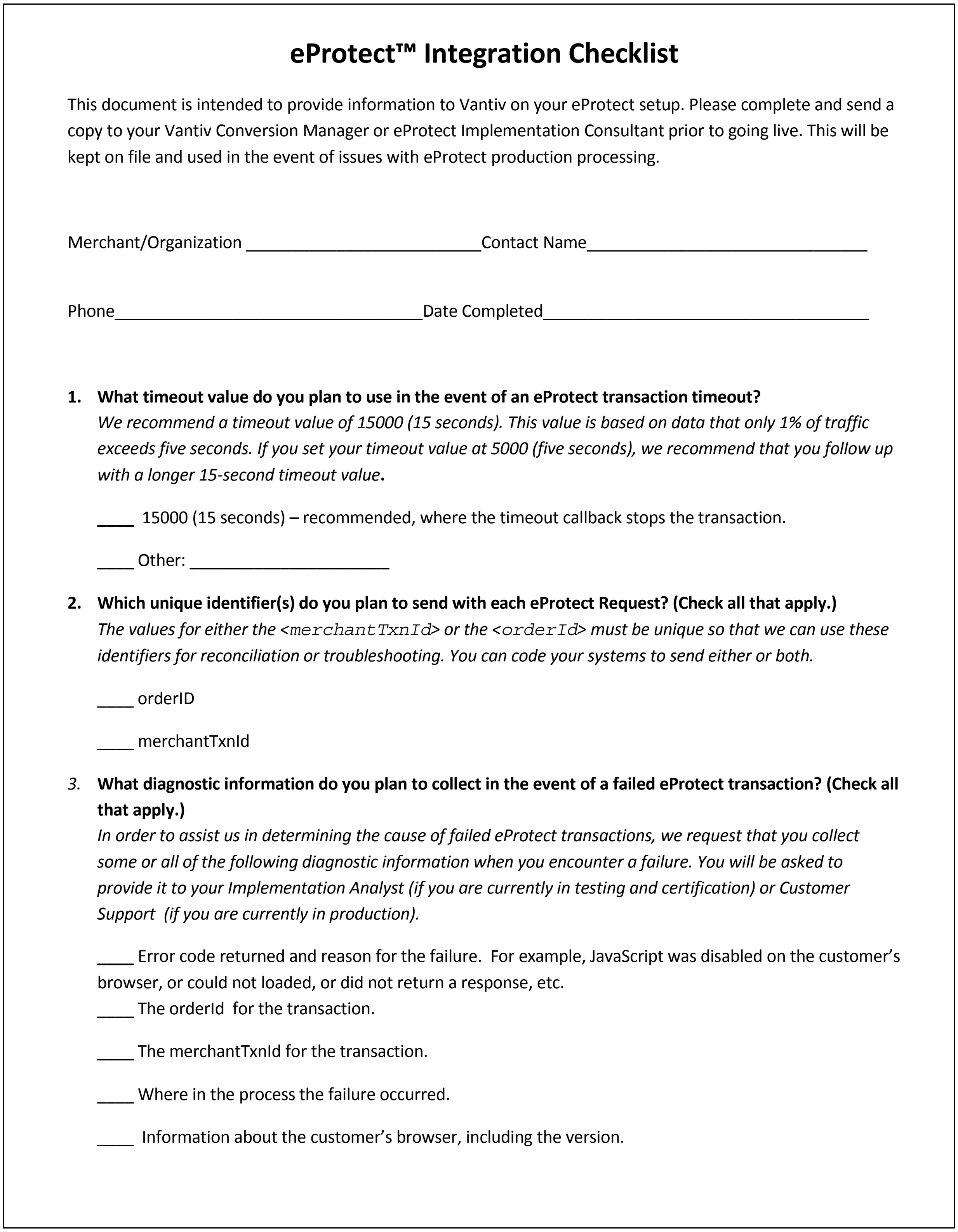
Property Name	Why excluded from white list?
background	The other properties of background like color or size can still be set with the more specific properties
background-attachment	Only makes sense in the context of background-image
background-image	Allows URL
background-origin	Only makes sense in the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

C

SAMPLE EPROTECT INTEGRATION CHECKLIST

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

FIGURE C-1 Sample eProtect Integration Checklist



eProtect™ Integration Checklist

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

Merchant/Organization ___________________________Contact Name________________________________

Phone___________________________________Date Completed______________________________________

1. **What timeout value do you plan to use in the event of an eProtect transaction timeout?**
*We recommend a timeout value of 15000 (15 seconds). This value is based on data that only 1% of traffic exceeds five seconds. If you set your timeout value at 5000 (five seconds), we recommend that you follow up with a longer 15-second timeout value***.**

____ 15000 (15 seconds) – recommended, where the timeout callback stops the transaction.

____ Other: _______________________

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

____ orderID

____ merchantTxnId

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

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

____ The merchantTxnId for the transaction.

____ Where in the process the failure occurred.

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

Index