

# eProtect™ Integration Guide - Enterprise

January 2018

cnpAPI Release V12.0

eProtect API V3.0

Document Version: 2.7

Vantiv eProtect™ Integration Guide Document Version: 2.7

All information whether text or graphics, contained in this manual is confidential and proprietary information of Vantiv, LLC and is provided to you solely for the purpose of assisting you in using a Vantiv, LLC product. All such information is protected by copyright laws and international treaties. No part of this manual may be reproduced or transmitted in any form or by any means, electronic, mechanical or otherwise for any purpose without the express written permission of Vantiv, LLC. The possession, viewing, or use of the information contained in this manual does not transfer any intellectual property rights or grant a license to use this information or any software application referred to herein for any purpose other than that for which it was provided. Information in this manual is presented "as is" and neither Vantiv, LLC or any other party assumes responsibility for typographical errors, technical errors, or other inaccuracies contained in this document. This manual is subject to change without notice and does not represent a commitment on the part Vantiv, LLC or any other party. Vantiv, LLC does not warrant that the information contained herein is accurate or complete.

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

Copyright © 2003-2018, Vantiv, LLC - ALL RIGHTS RESERVED.

---

# CONTENTS

---

## About This Guide

Intended Audience .....	v
Revision History .....	v
Document Structure .....	xi
Documentation Set .....	xi
Typographical Conventions .....	xii
Contact Information.....	xii

## Chapter 1 Introduction

eProtect Overview.....	2
How eProtect Works .....	4
Getting Started with eProtect.....	6
Migrating From Previous Versions of the eProtect API.....	6
From eProtect with jQuery 1.4.2 .....	6
From JavaScript Browser API to iFrame .....	7
Browser and Mobile Operating System Compatibility .....	8
Communication Protocol Requirement .....	8
eProtect Support for Apple Pay™ / Apple Pay on the Web .....	8
eProtect Support for Android Pay™ .....	9
eProtect Support for Pay with Google™ .....	10
eProtect Support for Visa Checkout™ .....	10
Getting Started with Visa Checkout .....	10
Requirements for Using Visa Checkout .....	11
jQuery Version .....	12
Certification and Testing Environment .....	12
Pre-Live Environment Limitations and Maintenance Schedule .....	13
Transitioning from Certification to Production .....	14
eProtect-Specific Response Codes .....	14
eProtect Registration ID Duplicate Detection .....	15
Creating a Customized CSS for iFrame.....	16
CSS iFrame Validation and Customization Features.....	16
Using Web Developer Tools .....	20
Reviewing your CSS with Vantiv .....	20

## Chapter 2 Integration and Testing

Integrating Customer Browser JavaScript API Into Your Checkout Page .....	24
Integration Steps .....	24
Loading the eProtect API and jQuery.....	25

Specifying the eProtect API Request Fields .....	26
Specifying the eProtect API Response Fields.....	27
Handling the Mouse Click .....	27
Intercepting the Checkout Form Submission .....	29
Handling Callbacks for Success, Failure, and Timeout.....	30
Success Callbacks .....	30
Failure Callbacks.....	31
Timeout Callbacks.....	32
Detecting the Availability of the eProtect API.....	32
Using the Customer Browser JavaScript API for Apple Pay on the Web.....	33
Using the Customer Browser JavaScript API for Visa Checkout .....	35
Adding Visa Checkout to the eProtect Customer Browser JavaScript API .....	36
Requesting and Configuring the API Key and externalClientId.....	36
Sending Vantiv the Required Fields.....	37
Integrating iFrame into your Checkout Page .....	38
Integration Steps .....	38
Loading the iFrame .....	38
Configuring the iFrame.....	39
Calling the iFrame for the Registration ID .....	41
Calling the iFrame for the Checkout ID .....	42
Notes on the PCI Non-Sensitive Value Feature .....	43
Handling Callbacks .....	43
Handling Errors .....	44
Integrating eProtect Into Your Mobile Application.....	46
Creating the POST Request .....	46
Sample Request.....	47
Sample Response .....	48
Using the Vantiv Mobile API for Apple Pay .....	48
Creating a POST Request for an Apple Pay Transaction .....	51
Sample Apple Pay POST Request .....	52
Sample Apple Pay POST Response.....	53
Using the Vantiv Mobile API for Visa Checkout .....	53
Sending Vantiv the Required Fields .....	55
Sample Visa Checkout POST Request.....	55
Sample Visa Checkout POST Response .....	56
Using the Vantiv Mobile API for Android Pay.....	56
Using the Vantiv Mobile API for Pay with Google .....	59
Collecting Diagnostic Information .....	62
Transaction Examples When Using ISO 8583, 610, HHMI, and PWS .....	63
Vantiv Online Systems - Acquirer ISO 8583 Message Format .....	64
Vantiv Online Systems - 610 Message Format.....	67

Response .....	68
(HHMI) Host-to-Host Format .....	70
Payment Web Services (PWS) External Payments .....	71
Transaction Examples When Using cnpAPI .....	75
Transaction Types and Examples .....	75
Authorization Transactions.....	76
Authorization Request Structure .....	76
Authorization Response Structure .....	77
Sale Transactions .....	79
Sale Request Structure .....	79
Sale Response Structure .....	80
Register Token Transactions .....	82
Register Token Request .....	82
Register Token Response.....	83
Force Capture Transactions.....	84
Force Capture Request.....	84
Force Capture Response .....	85
Capture Given Auth Transactions .....	86
Capture Given Auth Request .....	86
Capture Given Auth Response .....	88
Credit Transactions .....	89
Credit Request Transaction .....	89
Credit Response .....	90
Testing and Certification .....	92
Testing eProtect Transactions .....	93

## Appendix A Code Samples and Other Information

HTML Checkout Page Examples.....	98
HTML Example for Non-eProtect Checkout Page .....	98
HTML Example for JavaScript API-Integrated Checkout Page.....	99
HTML Example for Hosted iFrame-Integrated Checkout Page.....	102
Information Sent to Order Processing Systems.....	106
Information Sent Without Integrating eProtect .....	106
Information Sent with Browser-Based eProtect Integration .....	106
Information Sent with Mobile API-Based Application Integration .....	107
cnpAPI Elements for eProtect.....	108
cardValidationNum.....	109
checkoutId.....	110
expDate.....	111
paypage .....	112
paypageRegistrationId .....	113

registerTokenRequest.....	114
registerTokenResponse.....	115
token .....	116

## **Appendix B CSS Properties for iFrame API**

CSS Property Groups .....	118
Properties Excluded From White List.....	132

## **Appendix C Sample eProtect Integration Checklist**

## **Index**

# ABOUT THIS GUIDE

This guide provides information on integrating eProtect, which, when used together with Omnitoken, will reduce your exposure to sensitive cardholder data and significantly reduce your risk of payment data theft. 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 Release	All
1.1	Updated with revised URL for sample POST.	2.2

**TABLE 1** Document Revision History (Continued)

Doc. Version	Description	Location(s)
1.2	Updated to include Vantiv-Hosted PayPage iFrame solution (many changes and re-arrangement of sections). Also includes addition of Appendix B, and new HTML and JavaScript samples in Appendix A, <i>PayPage iFrame Solution</i> (many changes and re-arrangement of sections).	All
	'Migrating From Previous Versions of the PayPage API.'	
	'PayPage Certification, Testing, and Production URLs for mobile POST.'	
	Example for credit response code mappings for ISO 8583.	
1.3	Support for PCI Non-Sensitive values through PayPage.	Sections 2.1.5, 2.2.4, 2.2.5, 2.3.1, 2.3.1.1
	Updating Handling Callbacks section for handling errors differently depending on whether they are user error (recoverable) and non-user error (non-recoverable).	
1.4	Added link for iFrame CSS demo site.	Sections 1.3.6, 1.3.7, 2.2.3, 2.2.5, 2.3.1
	Added PayPage iFrame-Specific Response Codes.	
	Added new optional Common Property parameters for configuring iFrame.	
	Updated code to reflect revisions.	
	Updated information on testing URLs and User Agent examples.	
1.5	Added new Native Applications on Mobile Operating Systems.	Sections 1.3.2, 1.3.1.2
	Added step for JavaScript Browser API to Vantiv-Hosted iFrame on creating cascade style sheet.	
1.6	Updated product name in applicable areas throughout the guide from <i>PayPage</i> to <i>eProtect</i> (Phase 1). Updated many instances of <i>PayFrame</i> to <i>iFrame</i> .	ALL, Sections 1.1, 1.3.4, Documentation Set, Chapter 2
	Updated the eProtect workflow diagrams and steps.	
	Change term to <i>iFrame</i> from <i>Vantiv-Hosted</i> .	
	Update with applicable support & contact information.	
	1.3.4 - Corrected iFrame link in Certification and Testing Environments.	
	Added Online Developer Platform (ODP).	



**TABLE 1** Document Revision History (Continued)

Doc. Version	Description	Location(s)
1.7	Added information on obtaining CSS sample files from Vantiv.	Chapter 1, Appendix B
	Increased recommended transaction timeout value to 15000 (15 seconds) from 5000 (five seconds).	Chapter 2, Appendix A and C
	'CC Num - Token Generated' field has '1' value replaced with '&#' to avoid PCI sensitive values within document.	Appendix A, Section 1.3
1.8	Updated the paypageRegistrationId value to '1111222233330001.'	Section 2.3.1.2
	Added information on using the pciNonSensitive value in the iFrame and POST requests.	Sections 2.2.4 and 2.3.1
1.9	Added information on maximum length and data type for <code>orderId</code> and <code>id</code> parameters.	Sections 2.1.3, 2.2.4, and 2.3.1
1.10	Added additional information on the PCI non-sensitive value.	Section 2.2.4.
1.11	Re-arranged the example section and added a note related to eProtect and PWS (only available to existing PWS customers).	Section 2.6
	Removed the sample JavaScripts in Appendix A. Sample scripts are available at the pre-live, post-live, and production eProtect URLs.	Section 1.3.4, Appendix A.3
1.12	Added information on LitleXML elements, example transactions, response codes used for eProtect.	Section 1.3, 2.6, 2.7, Appendix A
	Added notes to LitleXML transaction examples related to expiration dates (required for eProtect transactions).	Section 2.7
1.13	Added information on loading the JavaScript API (must be loaded daily to your checkout page).	Chapter 2
1.14	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)
1.15	Added information on API V3 JavaScript (updated URLs and code examples); updated the XML examples to reference LitleXML 11.0.	Chapter 1 and 2
	Updated testing information with card numbers; added information on testing on multiple devices and browsers.	Chapter 2
	Updated the descriptions of <code>id</code> and <code>orderId</code> fields/elements.	Chapter 2
	Added a table of TPS Response Codes to the example section for the 610 messaging format.	Chapter 2
	Added recommendation on avoiding 'Flash of Un-styled Content' (FOUC) issues.	Chapter 2
1.16	Added information on new support for Apple Pay and Android Pay.	Chapter 1 and 2
2.0	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
	Added additional information on using the eCommerce option in 610 messages.	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.; also changed instances of 'LitleXML' to '' due to change in product name.	All
	Updated cnpAPI version to 11.1.	Chapter 2
2.1	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
	Added information on support for Visa Checkout™. <b>Note:</b> the new sections are for information purposes only, as Visa Checkout is not generally available with this release.	Chapter 1 and Chapter 2
2.2	Updated the <i>testlitle.com</i> sample page URLs to <i>testvantivcnp.com</i> . Added further information on which sample page URL to view when using the new enhanced iFrame features.	Chapter 1

**TABLE 1** Document Revision History (Continued)

Doc. Version	Description	Location(s)
2.3	Added information on the new CVV low-value token (checkoutId) feature for iFrame and JS Customer Browser API, including new response codes, checkoutIdMode and the getCheckoutId function.	Chapter 1 and 2
	Added information on the new cnpAPI element, <checkoutId>.	Appendix A
	Removed information and the URL for post-live regression testing as it is not applicable to Enterprise merchants.	Chapter 1
	Corrected the Request Submission URL in Table 1-2 from <i>https://request-eprotect.vantivprelive.com</i> to <i>https://request.eprotect.vantivprelive.com</i> .	Chapter 1
	Added information on Pre-Live Certification Environment maintenance and limitations.	Chapter 1
	Removed notes on non-general availability for Visa Checkout; it is now generally available to all merchants.	Chapter 1 and 2
	Added information on eProtect Registration ID Duplicate Detection.	Chapter 1
2.4	Added information on required communication protocol.	Chapter 1
	Corrected some instances of 'eProtect' in code samples.	Appendix A
2.5	Minor correction to code examples in section A.1.3.	Appendix A
2.6	Updated all cnpAPI element names to replace <i>Little</i> with <i>cnp</i> . For example, <i>littleToken</i> is now <i>cnpToken</i> , <i>littleOnlineRequest</i> is now <i>cnpOnlineRequest</i> , etc. This includes the namespace: <i>http://www.little.com/schema</i> is now <i>http://www.vantivcnp.com/schema</i> .	Chapter 2
	Corrected various eProtect element spelling errors, cleaned up miscellaneous coding in the HTML examples.	Chapter 2 and Appendix A
	Re-worked Section 1.3.9, Creating a Customized CSS for iFrame for better clarification.	Chapter 1
	Added information on new CSS-allowed Appearance properties.	Appendix B

**TABLE 1** Document Revision History (Continued)

Doc. Version	Description	Location(s)
2.7	Re-arranged and relocated the <i>Creating a Customized CSS for iFrame</i> 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, 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

For additional information, see the following Vantiv documentation:

- *Acquirer ISO 8583 Message Format* (Effective 02.15.2017)
- *Vantiv Online Systems 610 Interface Reference Guide* (Effective 02.15.2017)
- *HHMI: Host-To-Host Format Specification* (Effective Date 02.15.2017)

""

---

---

**NOTE:** Using eProtect in combination with PWS is available to existing PWS customers only.

---

---

- *Payment Web Services (PWS) External Payments Developers' Guide* (V6.1.2)
  - *Payment Web Services (PWS) Release Notes v6.1.2 and higher*
  - *Payment Web Services (PWS) Frequently Asked Questions*
- *Vantiv cnpAPI Reference Guide*

# Typographical Conventions

Table 2 describes the conventions used in this guide.

**TABLE 2** Typographical Conventions

Convention	Meaning
. . . . . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
< >	Angle brackets are used in the following situations: <ul style="list-style-type: none"><li>• user-supplied values (variables)</li><li>• XML elements</li></ul>
[ ]	Brackets enclose optional clauses from which you can choose one or more option.
<b>bold text</b>	Bold text indicates emphasis.
<i>Italicized text</i>	Italic type in text indicates a term defined in the text, the glossary, or in both locations.
<a href="#">blue text</a>	Blue text indicates a hypertext link.

## Contact Information

This section provides contact information for organizations within Vantiv.

**Implementation** - For technical assistance related to eProtect issues encountered during the testing and certification process, please contact your assigned Vantiv Conversion Manager or eProtect Implementation Consultant. If you do not have an assigned Implementation Consultant, please contact your Relationship Manager.

**Vantiv Contact Center** - For technical issues related to eProtect in production.

<b>Contact</b>	1-866-622-2390
<b>Hours Available</b>	24/7 (seven days a week, 24 hours a day)

**Relationship Management**- For non-technical issues, please contact your Vantiv Relationship Manager.

---

# 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™](#)
  - [eProtect Support for Pay with Google™](#)
  - [eProtect Support for Visa Checkout™](#)
- [Creating a Customized CSS for iFrame](#)

## 1.1 eProtect Overview

Vantiv's eProtect and OmniToken solutions help solve your card-not-present challenges by virtually eliminating payment data from your order handling systems, transferring the ownership to Vantiv and reducing PCI applicable controls.

The eProtect feature controls the fields on your checkout page that hold sensitive card data. When the cardholder submits their account information, your checkout page calls the eProtect JavaScript to register the provided credit card for a token. The JavaScript API 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.

Vantiv ensures high service availability for eProtect by implementing primary and secondary endpoints (i.e., routing to a secondary site if the primary site is unavailable). High availability is supported when using the eProtect JavaScript API V.3; it is not available for the Mobile API option.

Vantiv provides three integration options for eProtect:

- **iFrame API** - this solution builds on the same architecture of risk- and PCI 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 CVV2 values are hosted from our PCI-Compliance environment, rather than embedded as code into your checkout page within your environment.
- **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.
- **Mobile API** - eProtect Mobile Native Application allows you to use the eProtect solution to handle payments without interacting with the eProtect JavaScript in a browser. With Mobile Native Application, 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--to replace payment card data from your web servers.

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.

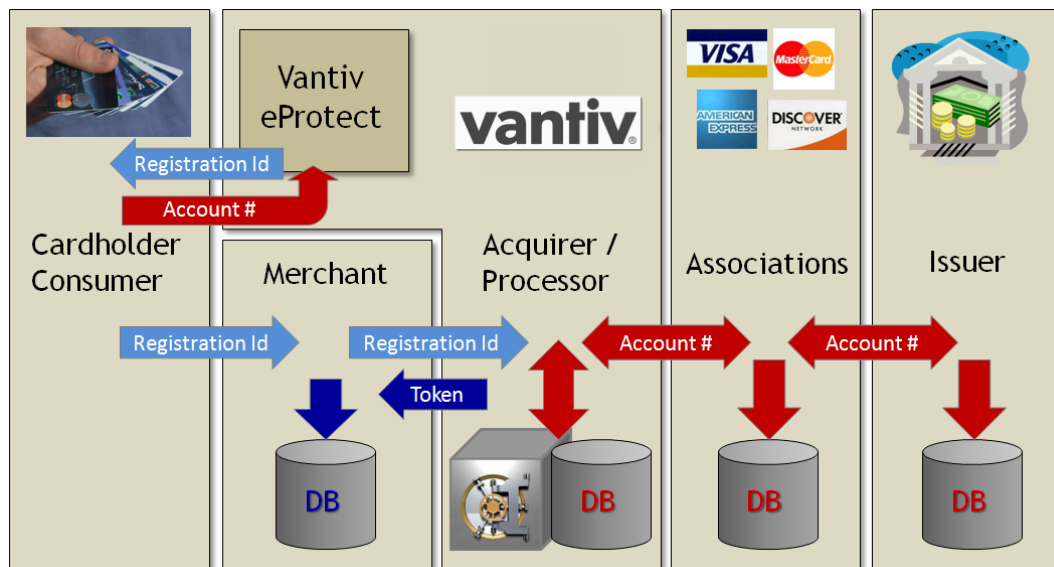
Figure 1-1 illustrates eProtect with Omnitoken. See the section, [How eProtect Works](#) on page 4 for additional details.

---

**NOTE:** In order to optimally use the eProtect feature for risk reduction, this feature must be used at all times, without exception.

---



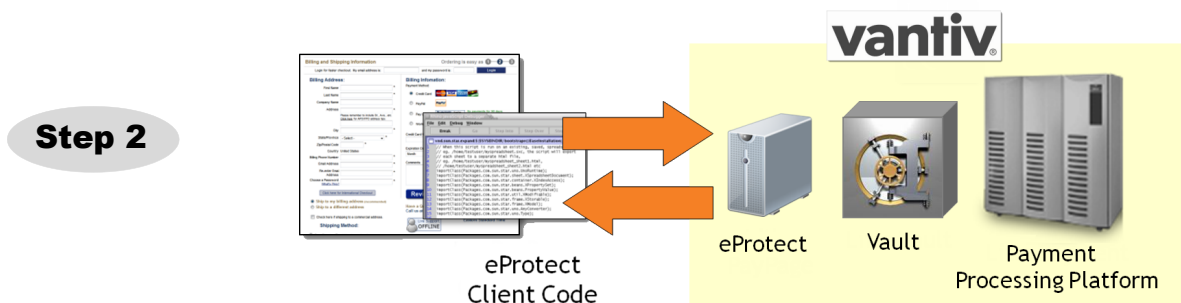
**FIGURE 1-1** eProtect

## 1.2 How eProtect Works

This section illustrates the eProtect process.

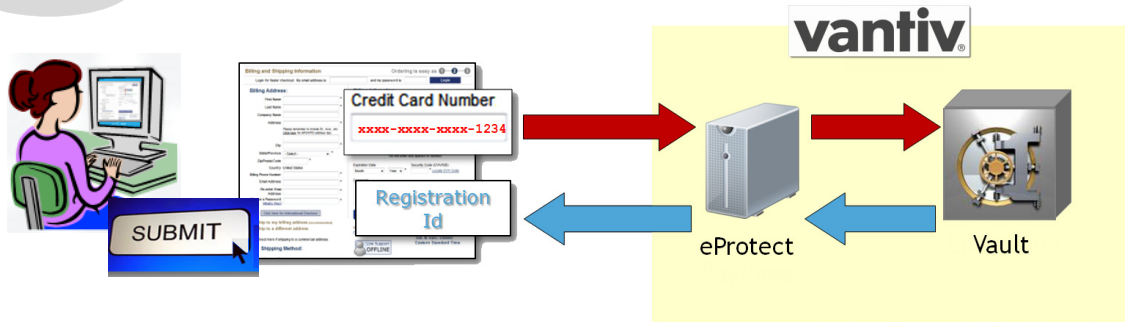


1. When your customer is ready to finalize a purchase from your website or mobile application, your web server delivers your Checkout form to the customer's web browser or mobile device.

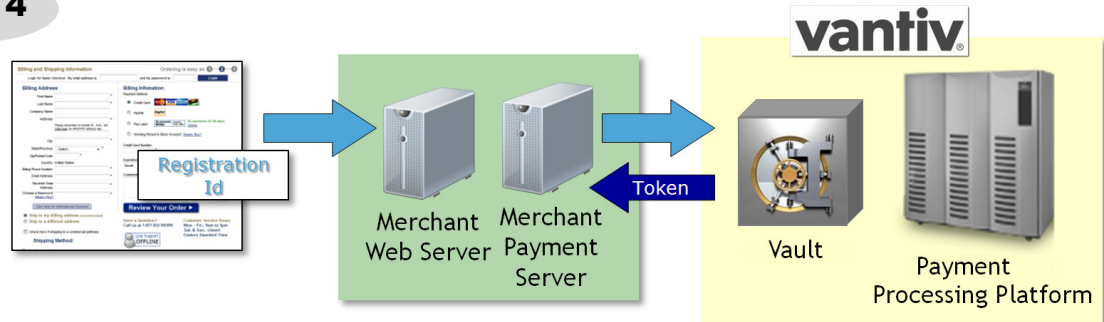


2. If using the iFrame API, the browser loads the iFrame URL hosted by the eProtect server. If using the JavaScript API, the browser loads the eProtect Client code (JavaScript) from the eProtect server. The API validates credit cards, submits account numbers to the eProtect Service, encrypts account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

(continued on next page)

**Step 3**

3. After the customer enters their card number and clicks or taps the submit button, the eProtect APIs sends the card number data to our eProtect service which calls our Security Services vault or submits a cnpAPI transaction to register an OmniToken for the card number provided. The token is securely stored in Enterprise Security Services for 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 for web processing, or to their mobile device as a POST response.

**Step 4**

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 Enterprise Security Services vault maps the Registration ID to the OmniToken and card number, processing the payment as usual. The response message from the applicable Vantiv front end contains the OmniToken value, which you store as you would a credit card.

## 1.3 Getting Started with eProtect

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

- Ensure that your organization is enabled and certified to process OmniTokens, using the OmniToken solution.
- Complete and return 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 your eProtect Implementation Consultant.
- 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 16 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 24
  - [Integrating iFrame into your Checkout Page](#) on page 38
  - [Integrating eProtect Into Your Mobile Application](#) on page 46 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 [Transaction Examples When Using ISO 8583, 610, HHMI, and PWS](#) on page 63 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 102.

1. Remove the script that was downloading `eProtect-api3.js`.
2. Add a script tag to download `eprotect-iframe-client3.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 16 and [Configuring the iFrame](#) on page 39 for more information.
8. See [Calling the iFrame for the Registration ID](#) on page 41 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 <i>Paired with iPhone 6 and later</i>	Watch OS 3 and later	
iMac <i>Paired with any of the above mobile devices with ID Touch for authentication</i>	macOS Sierra and later	
MacBook <i>Paired with any of the above mobile devices with ID Touch for authentication</i>	macOS Sierra and later	

### 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 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 Android Pay](#) on page 56 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 59 for more information.

### 1.3.6 eProtect Support for Visa Checkout™

Visa Checkout™ is a digital payment service in which consumers can store card information for Visa, MasterCard, Discover, and American Express cards. Visa Checkout provides quick integration for merchants that want to accept payments from these card holders. Visa Checkout is flexible and imposes only a few requirements for its use, leveraging your existing environments--web site and mobile applications--where you add Visa Checkout buttons to existing pages and implement business and event logic using programming languages, tools, and techniques in the same way you currently do. Vantiv supports Visa Checkout purchases from your website or mobile app. initiated from compatible devices.

#### 1.3.6.1 Getting Started with Visa Checkout

---

**NOTE:** Parts of this section are excerpts from Visa Checkout documentation and represents data available at the time of publication of this document, and is therefore subject to change. See the latest Visa documentation ([https://developer.visa.com/products/visa\\_checkout/reference](https://developer.visa.com/products/visa_checkout/reference)) for current information.

---



The simplest approach to integrating Visa Checkout takes three steps and can be done entirely from your web page (with the exception of decrypting the consumer information payload on a secure server). [Figure 1-2](#) illustrates the main steps for getting started with Visa Checkout.

**FIGURE 1-2** Getting Started with Visa Checkout



1. Place a Visa Checkout button on your web page and include the necessary JavaScript to handle events associated with the button.
2. Handle the payment event returned by Visa Checkout by decrypting the consumer information payload.
3. Update Visa Checkout with the final payment information after the payment has been processed.

All integration options require that you perform step 1. Sections in this document describes the method using Vantiv eProtect.

### 1.3.6.2 Requirements for Using Visa Checkout

This section describes the various requirements for using Visa Checkout.

- **Usage and Placement of Visa Checkout Buttons:** You are required to implement the Visa Checkout branding requirements on all pages where the consumer is presented payment method options, such as Visa Checkout or another payment method. Common examples include shopping cart page, login page, product page, and payment page. Your actual implementation depends on your specific flow.

You can use Visa Checkout on any web page or in any flow on your site or native mobile application where a consumer is asked to type in their billing and payment information. Common examples include cart pages (both full and mini) pages, payment pages, card-on-file management pages, or immediately before a flow where a consumer is prompted for personal information, which may be available, at least partially, within Visa Checkout.

See the latest *Visa Checkout Integration Guide* for more information and to learn how placing Visa Checkout buttons on the shopping cart page and your login page might work.

- **Clickjacking Prevention Steps:** To prevent clickjacking of your pages, each page must contain JavaScript to verify that there are no transparent layers, such as might be the case if your page was loaded as an iFrame of a page containing malicious code, and that only your site can load your pages.

See the latest *Visa Checkout Integration Guide* for more information on preventing clickjacking.

- **Obtaining the externalClientId from Vantiv:** During the on-boarding process, Vantiv Implementation assigns an `externalClientId` to denote the relationship between Vantiv, your organization and Visa.
- **Updating Visa Checkout with the Payment Information:** After you finish making a payment (and perhaps using the information from the payload), you must update the payment information in Visa Checkout. To update Visa Checkout from a Thank You page (next page to load after making the payment), you add a one-pixel image to the page.

For more information about the Update Payment Info pixel image, see the latest *Visa Checkout Integration Guide*.

See additional information on [Using the Customer Browser JavaScript API for Visa Checkout](#) on page 35 and [Using the Vantiv Mobile API for Visa Checkout](#) on page 53.

### 1.3.7 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.

### 1.3.8 Certification and Testing Environment

For certification and testing of Vantiv feature functionality, Vantiv uses the **Pre-Live** testing environment. 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.

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

- eProtect JavaScript (eProtect-api3.js)
- iFrame Client (eprotect-iframe-client3.js)
- iFrame JavaScript (eProtect-iframe.js)

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

Environment	URL Purpose	URL
Pre-Live (Testing and Certification)	JavaScript Library	https://request.eprotect.vantivprelive.com/eProtect/eProtect-api3.js
	Request Submission (excluding POST)	https://request.eprotect.vantivprelive.com
	iFrame	https://request.eprotect.vantivprelive.com/eProtect/js/eProtect-iframe-client3.min.js
	POST Request Submission (for Mobile API)	https://request.eprotect.vantivprelive.com/eProtect/paypage
	API Key Request (Visa Checkout only)	https://request.eprotect.vantivprelive.com/eProtect/keys.json
Live Production	<i>Production</i>	<i>Contact your Implementation Consultant for the eProtect Production URL.</i>

### 1.3.8.1 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 - every other Thursday from 10:00 PM to 6:00 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.

---

### 1.3.9 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 the above table and in the sample scripts throughout this guide should only be used in the certification and testing environment.**

### 1.3.10 eProtect-Specific Response Codes

[Table 1-3](#) lists response codes specific to the eProtect feature, received in the browser or mobile device, and those received via the applicable Vantiv message specification responses. [Table 1-4](#) lists those received via a cnpAPI Response. For further information on response codes specific to token transactions, see the publications listed in [Documentation Set](#) on page xi.

**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
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

---

**NOTE:** For information on response codes specific to OmniToken transactions, see the applicable Vantiv message interface specification.

---

**TABLE 1-4** eProtect Response Codes Received in cnpAPI Responses

Response Code	Response Message	Response Type	Description
826	Checkout ID was invalid	Soft Decline	An eProtect response indicating that the Checkout ID submitted was too long, too short, non-numeric, etc.
827	Checkout ID was not found	Soft Decline	An eProtect response indicating that the Checkout ID submitted was expired, or valid but not found.
828	Generic Checkout ID error	Soft Decline	There is an unspecified Checkout ID error; contact your Relationship Manager.
877	Invalid PayPage Registration ID	Hard Decline	An eProtect response indicating that the Registration ID submitted is invalid.
878	Expired PayPage Registration ID	Hard Decline	An eProtect response indicating that the Registration ID has expired (Registration IDs expire 24 hours after being issued).
879	Merchant is not authorized for PayPage	Hard Decline	A response indicating that your organization is not enabled for the eProtect feature.

### 1.3.11 eProtect Registration ID Duplicate Detection

Vantiv performs duplicate detection reviews on all form fields such as the card number, CVV, order ID, and Transaction ID. In the event that multiple eProtect registrations are submitted within a five-minute period containing identical form fields, subsequent requests are flagged as duplicates, and processed by returning the originating callback response and Registration ID value.

With this, false positives may occur if your organization has not implemented a policy that provides a unique Order ID and Transaction ID for every request. If not implemented, you could potentially receive an incorrect CVV value, which could be disruptive to chargeback processing. Vantiv strongly recommends that the order ID and transaction ID data elements be unique for every registration request.

## 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.

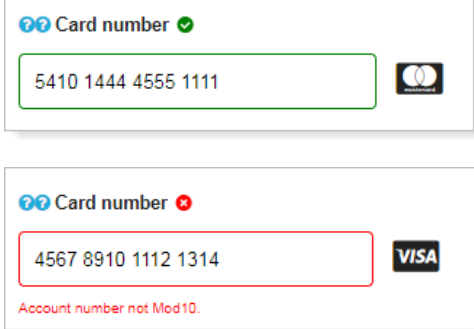
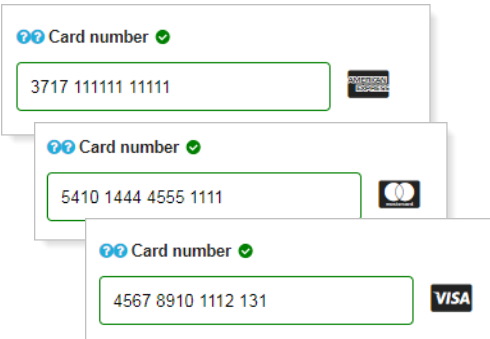
**Client Driven Behaviors** - additional capabilities include:

- *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.

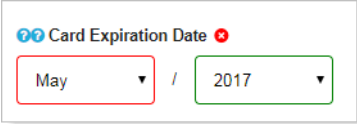
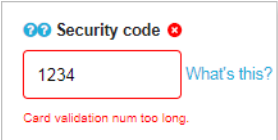
Table 1-5, "iFrame Checkout Page Customizations - In-Line Field Validations" and Table 1-6, "Style Sheet and iFrame Customizations" show samples of these CSS iFrame customizations and describes the implementation of each.

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

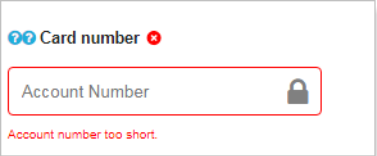
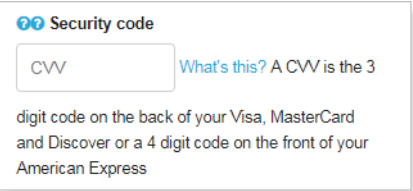
Field	Validation Behavior	Samples
Card Number	<p>The iFrame checks the card number for correct size (too short or too long) and against the Luhn/Mod10 algorithm.</p> <p>In this example, if the consumer's inputs are valid, you can configure the iFrame to display green field borders and include a green check mark. Red borders and a red 'X' can indicate invalid input.</p> <p>The error messages and frame colors are customizable in your style sheet.</p>	
	<p>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.</p> <p>You can configure your style sheet to hide the card graphic.</p> <p>In addition, the iFrame auto-formats the arrangement of the card digits based on the initial entry.</p>	

**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.	
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.	

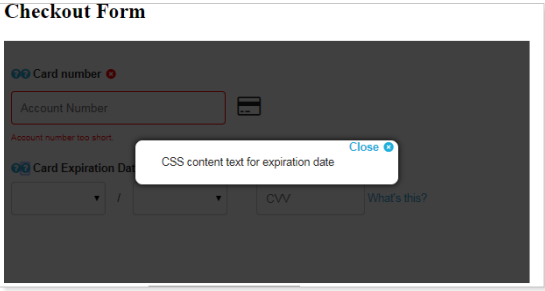
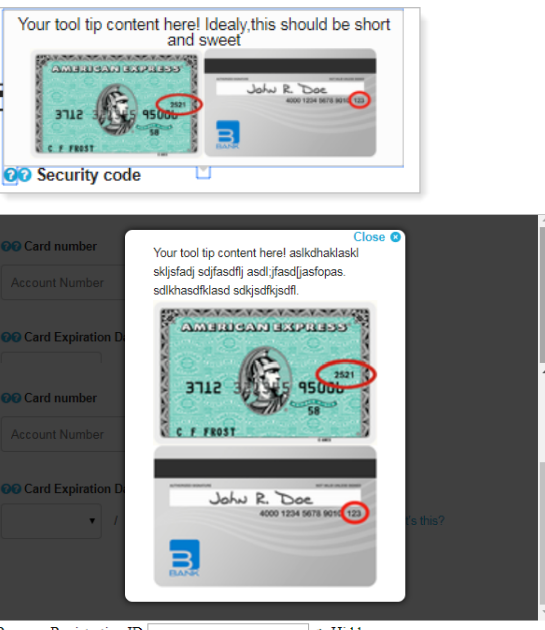
The items listed in [Table 1-6](#) are also available as optional features controlled by the your style sheet and via iFrame function. By default, the Tool Tip features are active, but can be suppressed with the CSS.

**TABLE 1-6** Style Sheet and iFrame Customizations

Customization	Samples
<b>Trust Badge</b> - 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 <i>in place of</i> the card graphic; your page cannot display both.	
<b>Tool Tips</b> - 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?'	 <p><i>Tool tip displayed after clicking 'What's This?'</i></p>



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

Customization	Samples
<p>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.</p> <p>You can also configure a modal dialog to activate on the click of the second ‘?’ icon to display more lengthy CSS content.</p>	<p><b>Checkout Form</b></p>  <p><i>Modal dialog displayed upon clicking second ‘?’ icon.</i></p>
<p><b>Tool Tips (continued)</b></p> <p>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.</p> <p>You can also remove the scrollbars, as well as direct your CSS to auto-size the dialog based on content.</p>	 <p><i>Modal dialog displayed upon clicking first or second ‘?’ icon at the security code field.</i></p>

## 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 16, use the following URL:

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



---

## 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 ISO 8583, 610, HHMI, and PWS](#)
- [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 following payment methods:

- [Using the Customer Browser JavaScript API for Apple Pay on the Web](#)
- [Using the Customer Browser JavaScript API for Visa Checkout](#)

See [Integrating eProtect Into Your Mobile Application](#) on page 46 for more information on the mobile solution.

See [Integrating iFrame into your Checkout Page](#) on page 38 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, Checkout 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 98.

## 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-api3.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 hidden request fields to your checkout form for `paypageId` (a unique number assigned by eProtect Implementation), `merchantTxnId`, `orderId`, and `reportGroup` (cnpAPI elements). Optionally, you can include the `checkoutId` when `CheckoutIdMode` is set to true. 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="checkoutId" name="checkoutId" 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
<b>ccNum</b>	(Optional) The credit card account number. Not applicable when <code>checkoutIdMode</code> is set to true.
<b>cvv2Num</b>	(Optional) The card validation number, either the CVV2 (Visa), CVC2 (MasterCard), or CID (American Express and Discover) value.
<b>paypageRegistrationId</b>	(Required) The temporary identifier used to facilitate the mapping of a token to a card number. Not applicable when <code>checkoutIdMode</code> is set to true.



**TABLE 2-1** eProtectFormFields Definitions

Field	Description
<b>checkoutId</b>	(Optional) The low-value token ID exchanged for the CVV value, when <code>checkoutIdMode</code> is set to true. (Checkout Id Mode can be used when you store the high-value token (Registration Id) on file for a consumer, but still want that consumer to populate the CVV with each eProtect transaction.)
<b>bin</b>	(Optional) The bank identification number (BIN), which is the first six digits of the credit card number. Not applicable when <code>checkoutIdMode</code> is set to true.

## 2.1.4 Specifying the eProtect API Response Fields

To specify the eProtect API Response fields, add hidden response fields on your checkout form for storing information returned by eProtect: `paypageRegistrationId`, `checkoutId`, `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$checkoutId" name="response$checkoutId"
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 an eProtect Request and calls `sendToEprotect`.

The `sendToEprotect` call includes a timeout value in milliseconds. If the response from the primary server takes more than five (5) seconds, the request is automatically sent to our

secondary server. To ensure the secondary server has time to respond, we recommend a timeout value of 15000 (15 seconds).

---

**NOTE:** The URL in this example script (**in red**) should only be used in the certification and testing environment. Before using your checkout page with eProtect 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,
            "checkoutIdMode": true
            "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
<b>paypageId</b>	(Required) The unique number assigned by Implementation.
<b>reportGroup</b>	(Required) The cnpAPI required attribute that defines under which merchant sub-group this transaction will be displayed in eCommerce iQ Reporting and Analytics.

**TABLE 2-2** eProtectRequest Fields (Continued)

Field	Description
<b>orderId</b>	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 <code>id</code> and <code>orderId</code> 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 <code>orderId</code> (and send it to your servers to map it to the order number that you generate later).
<b>id</b>	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 <code>id</code> and <code>orderId</code> must be different and unique so that we can use these identifiers for reconciliation or troubleshooting.
<b>checkoutIdMode</b>	(Optional) Determines whether <code>checkoutId</code> mode is activated. Setting the value to true causes only the <code>cvv2</code> form field to be exchanged with eProtect, and returns a <code>checkoutId</code> upon a successful callback (instead of the <code>paypageRegistrationId</code> ).
<b>applepay</b>	(Optional). The Apple Pay PKPaymentToken. Required for Apple Pay on the Web.
<b>url</b>	(Required) The URL to request submission for eProtect. See <a href="#">Table 1-2, eProtect Certification, Testing, and Production URLs</a> on page 13.

## 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$checkoutId').value = response.checkoutId;
    document.getElementById('response$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 14). 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 27. 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-api3.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)" );
        }
    }
...
</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

---

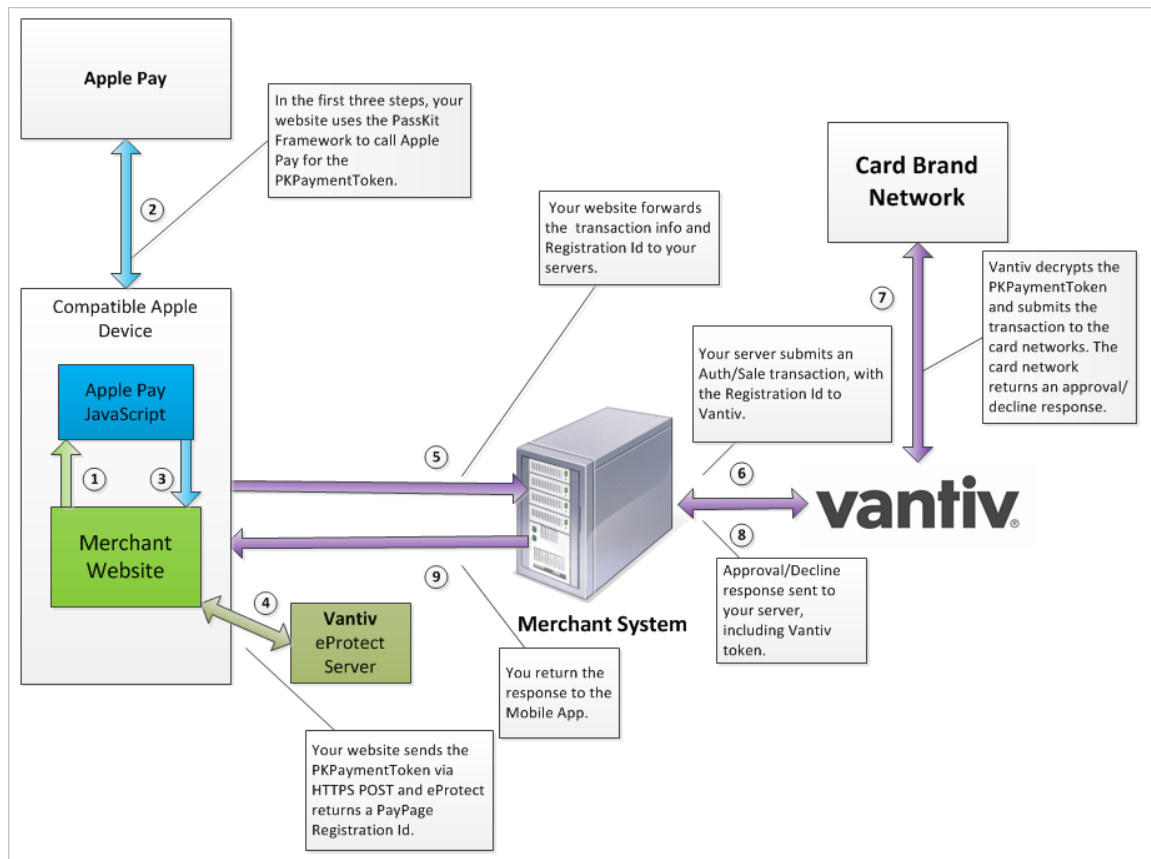
**NOTE:** This section is an excerpt from the Vantiv eCommerce Technical Publication, *Vantiv eCommerce Solution for Apple Pay*. Refer to the full document for further information.

---

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 2-3](#).

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



## 2.1.10 Using the Customer Browser JavaScript API for Visa Checkout

The operation of Visa Checkout is simple, but requires either the modification of your existing website or development of new website that include the use of the Visa Checkout SDK and handling of the encrypted data returned to your website by Visa Checkout. The basic steps that occur when a consumer initiates an Visa Checkout purchase using your website are:

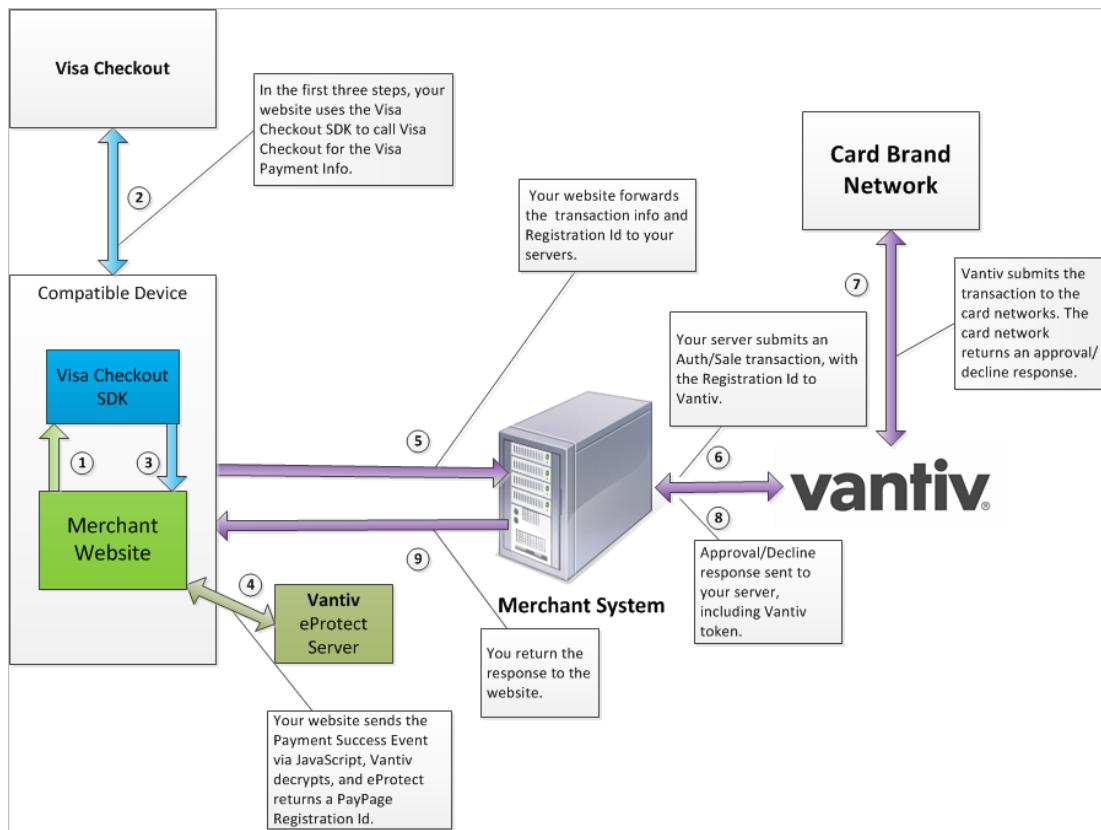
1. When the consumer selects the Visa Checkout option from your website, your site makes use of the Visa Checkout SDK to request payment data from Visa Checkout.
2. When Visa Checkout receives the call from your website, Visa creates a Payment Success event using the Vantiv API key. Included in the Payment Success event is encrypted PAN data.
3. Visa Checkout returns the Payment Success event (defined in Visa documentation; see [https://developer.visa.com/products/visa\\_checkout/guides](https://developer.visa.com/products/visa_checkout/guides)) to your website.

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 Visa Checkout button, communication is exchanged with Visa Checkout via the JavaScript API to obtain the Payment Success event.

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 transaction using that ID (outlined in the following steps)

4. Your website sends the Payment Success event to our secure server via the JavaScript Browser API and Vantiv decrypts the Payment Success event associated with the Registration ID. eProtect then returns a Registration ID along with customer information from the decrypted data.
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/submit an Authorization/Sale transaction using the Registration ID.
7. Vantiv 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.

After you finish making a payment, you update the payment information in Visa Checkout. To update Visa Checkout from a Thank You page (next page to load after making the payment), you add a one-pixel image to the page.

**FIGURE 2-2** Data/Transaction Flow - Vantiv Browser JavaScript API for Visa Checkout

### 2.1.11 Adding Visa Checkout to the eProtect Customer Browser JavaScript API

Integrating Visa Checkout into your web page includes the following:

- Requesting and Configuring the API Key and externalClientId
- Sending Vantiv the Required Fields

#### 2.1.11.1 Requesting and Configuring the API Key and externalClientId

To request and configure the API Key and externalClientId, insert the following JavaScript into your checkout page:

```
<script type="text/javascript"
src="https://request.eprotect.vantivprelive.com/eProtect/eProtect-api3.js"></script>
<script>
function onVisaCheckoutReady() {
    var ep = new eProtect();
```

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

```
V.init({
  apikey: ep.getVisaCheckoutApiKey(), //Vantiv's current Visa Checkout API Key
  sourceId: "Merchant Defined Source ID",
  externalClientId: "stefan_sandwiches", //Merchant's client id - get this
from Vantiv's implementations team
  settings: {
    ...
  },
  paymentRequest: {
    ...
  }
});
...
}
```

### 2.1.11.2 Sending Vantiv the Required Fields

Insert the following to send the required fields to Vantiv:

```
V.on("payment.success", function(payment){
  var eProtectRequest = {
    ...,
    "visaCheckout": payment
  };
  new eProtect().sendToEprotect(eProtectRequest, formFields, submitAfterEprotect,
onErrorAfterEprotect, timeout);
})
```

For an example of a completed checkout page with these components, go here:

<https://www.testvantivcn.com/checkout/checkout3VisaCheckout-prelive-sandbox.jsp>.

## 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 16 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 your 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 102.

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-  
client3.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`.

If you wish to prevent the occurrence of ‘Flash of Un-styled Content’ (FOUC), structure your code to load the iFrame and all related surrounding host page content in a hidden `div`. Once the iFrame reports it is ready, your site shows the whole `div`.

```
$( document ).ready(function() {  
  var configure = {  
    "paypageId":document.getElementById("request$paypageId").value,  
    "style":"test",  
    "reportGroup":document.getElementById("request$reportGroup").value,  
    "timeout":document.getElementById("request$timeout").value,  
    "div": "eProtectiframe",  
    "callback": eProtectiframeClientCallback,  
    "checkoutIdMode": true,  
    "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														
<b>paypageId</b>	(Required) The unique number assigned by Implementation.														
<b>style</b>	(Required) The CSS filename (excluding the '.css'). For example, if the style sheet filename is <code>mysheet1.css</code> , the value for this property is <code>mysheet1</code> .														
<b>reportGroup</b>	(Required) The <code>cnpAPI</code> required attribute that defines under which merchant sub-group this transaction will be displayed in eCommerce iQ Reporting and Analytics.														
<b>timeout</b>	(Required) The number of milliseconds before a transaction times out and the timeout callback is invoked. If the response from the primary server takes more than five (5) seconds, the request is automatically sent to our secondary server. To ensure the secondary server has time to respond, Vantiv recommends a timeout value of 15000 (15 seconds).														
<b>div</b>	(Required) The ID of the HTML <code>div</code> element where our iFrame is embedded as <code>innerHTML</code> .														
<b>callback</b>	<p>(Required) The function element that our iFrame calls with a single parameter representing a JSON dictionary. The keys in the callback are:</p> <table> <tr> <td>*paypageRegistrationId</td><td>*orderId</td></tr> <tr> <td>*bin</td><td>*response</td></tr> <tr> <td>*type</td><td>*responseTime</td></tr> <tr> <td>*firstSix</td><td>*message</td></tr> <tr> <td>*lastFour</td><td>*reportGroup</td></tr> <tr> <td>*expDate</td><td>*id</td></tr> <tr> <td>*vantivTxnId</td><td>*timeout</td></tr> </table>	*paypageRegistrationId	*orderId	*bin	*response	*type	*responseTime	*firstSix	*message	*lastFour	*reportGroup	*expDate	*id	*vantivTxnId	*timeout
*paypageRegistrationId	*orderId														
*bin	*response														
*type	*responseTime														
*firstSix	*message														
*lastFour	*reportGroup														
*expDate	*id														
*vantivTxnId	*timeout														
<b>checkoutIdMode</b>	(Optional) Determines whether <code>checkoutId</code> mode is activated. Set the value to <b>true</b> to establish a rule allowing the capture of the CVV only (in order to receive the <code>checkoutId</code> ). Adding this field hides all fields in the iFrame, except CVV and CVV-related fields (including tooltips, etc.).														
<b>inputsEmptyCallback</b>	(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 <a href="#">Creating a Customized CSS for iFrame on page 14</a> for more information.														

**TABLE 2-3** Common Properties (Continued)

Property	Description
<b>inlineFieldValidations</b>	(Optional) Determines whether in-field validations are performed (set value to <code>true</code> ). See <a href="#">Creating a Customized CSS for iFrame</a> on page 14 for more information.
<b>height</b>	(Optional) The height (in pixels) of the iFrame. There are three options: <ul style="list-style-type: none"><li>You can pass <code>height</code> as an optional parameter when configuring the client.</li><li>You can call <code>autoAdjustHeight</code> 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).</li><li>You can ignore <code>height</code>. The iFrame may display a vertical scroll bar, depending upon your styling of the <code>div</code> containing the iFrame.</li></ul>
<b>htmlTimeout</b>	(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 eProtect 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 CVV2. 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
        "pciNonSensitive": true
    };
    eProtectiframeClient.getPaypageRegistrationId(message);
    return false;
};
```

## 2.2.5 Calling the iFrame for the Checkout ID

Additionally, your checkout page can call the iFrame to exchange the CVV value for a checkoutId (low-value-token with a 24-hour lifespan). Use this when you store the high-value token (registrationId) on file for a consumer, but still want that consumer to populate the CVV with each eProtect transaction. See the parameters listed in [Table 2-4](#) for more information.

Note that the PCI non-sensitive flag is not applicable for the getCheckoutId function.

```
var message = {
  "id":document.getElementById("request$merchantTxnId").value,
  "orderId":document.getElementById("request$orderId").value
};

startTime = new Date().getTime();
iframeClient.getCheckoutId(message);
```

**TABLE 2-4** Event Handler Parameters

Parameter	Description
id	<p>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.</p> <p><b>Type:</b> String</p> <p><b>Max Length:</b> 25 characters</p> <p>Vantiv recommends that the values for <code>id</code> and <code>orderId</code> must be different and unique so that we can use these identifiers for reconciliation or troubleshooting.</p>
orderId	<p>The merchant-assigned unique value representing the order in your system (used when linking authorizations, captures, and refunds, and for retries).</p> <p><b>Type:</b> String</p> <p><b>Max Length:</b> 25 characters</p> <p>Vantiv recommends that the values for <code>id</code> and <code>orderId</code> 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 <code>orderId</code> (and send it to your servers to map it to the order number that you generate later).</p>
pciNonSensitive	<p>(Optional) Bypasses existing MOD10 validation for only non-sensitive cardholder data as defined by PCI (e.g. Private Label) for tokenization. A value of <code>true</code> bypasses MOD10 validation. See <a href="#">Notes on the PCI Non-Sensitive Value Feature</a>, next.</p> <p><b>Note:</b> If you use this parameter with a value of <code>true</code>, the card type and BIN are not returned in the response.</p>



### 2.2.5.1 Notes on the PCI Non-Sensitive Value Feature

eProtect is designed to capture branded credit cards from the major card networks including Visa, MasterCard, American Express, Discover, and JCB. The eProtect PCI Non-Sensitive feature has the capability to capture and tokenize non-network branded payment types, such as private label and Vantiv gift cards. The PCI Non-Sensitive feature can tokenize 13-19 digit values, as long as the Vantiv message interface specification supports that payment type.

Implementation of the `pciNonSensitive` parameter may require augmentation of your checkout page to include a method whereby your customer chooses a payment type (e.g., a drop-down box). You must capture the non-network branded payment type in order to send the appropriate flag when calling eProtect.

### 2.2.6 Handling Callbacks

After the iFrame has received the `paypageRegistrationId` or `checkoutId`, 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$vantivTxnId').value = response.vantivTxnId;
        document.getElementById('response$checkoutId').value = response.checkoutId;
        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
}
}
};

```

### 2.2.6.1 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 14.

- 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 Visa Checkout](#)
- [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: <b>request.eprotect.vantivprelive.com</b>	
	User-Agent = "User-Agent" ":" 1*( product   comment ) <i>For example: User-Agent: Vantiv/1.0 CFNetwork/459 Darwin/10.0.0.d3</i>	

**TABLE 2-5** POST Headers, Parameters, and URL (Continued)

Component	Element	Description
Parameters (required)	<b>paypageId</b>	The unique number assigned by Implementation.
	<b>reportGroup</b>	A unique value that you assign.
	<b>orderId</b>	A unique value that you assign (string, max length: 25 char.). See full definition on <a href="#">page 29</a> .
	<b>id</b>	A unique value that you assign (string, max length: 25 char.). See full definition on <a href="#">page 29</a> .
	<b>accountNumber</b>	A unique value that you assign. (Not used in Apple Pay transactions.)
	<b>pciNonSensitive</b>	Bypasses the existing MOD10 validation for only non-sensitive cardholder data as defined by PCI (e.g., Private Label) for tokenization. A value of true bypasses MOD10 validation. See <a href="#">Notes on the PCI Non-Sensitive Value Feature</a> on page 43.  <b>Note:</b> If you use this parameter with a value of <code>true</code> , the card type and BIN are not returned in the response.
	<b>cvv2</b>	The card validation number, either the CVV2 (Visa), CVC2 (MasterCard), or CID (American Express and Discover) value.
URL	<a href="https://request.eprotect.vantivprelive.com/eProtect/paypage">https://request.eprotect.vantivprelive.com/eProtect/paypage</a>	

**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&pciNonSensitive=true" https://request.eprotect.vantivprelive.com/eProtect/paypage
```

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

### 2.3.1.2 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
VGNFJoRmdNeXJUOH14Skl1TTQ1Z0t6WE9TYmdqdjBJT0F5N28zbUpxdlhGazZFdm1CSzdTN3ptKw\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
paypageRegistrationId	The temporary identifier used to facilitate the mapping of a token to a card number.
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.
orderId	(Mirrored back from the request) The merchant-assigned unique value representing the order in your system. <b>Type:</b> String <b>Max Length:</b> 25 characters
id	(Mirrored back from the request) The merchant-assigned unique value representing this transaction in your system. <b>Type:</b> String <b>Max Length:</b> 25 characters
vantivTxnId	The automatically-assigned unique transaction identifier.
firstSix	(Mirrored back from the request) The first six digits of the credit card number.
lastFour	(Mirrored back from the request) The last four digits of the credit card number.

## 2.3.2 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.

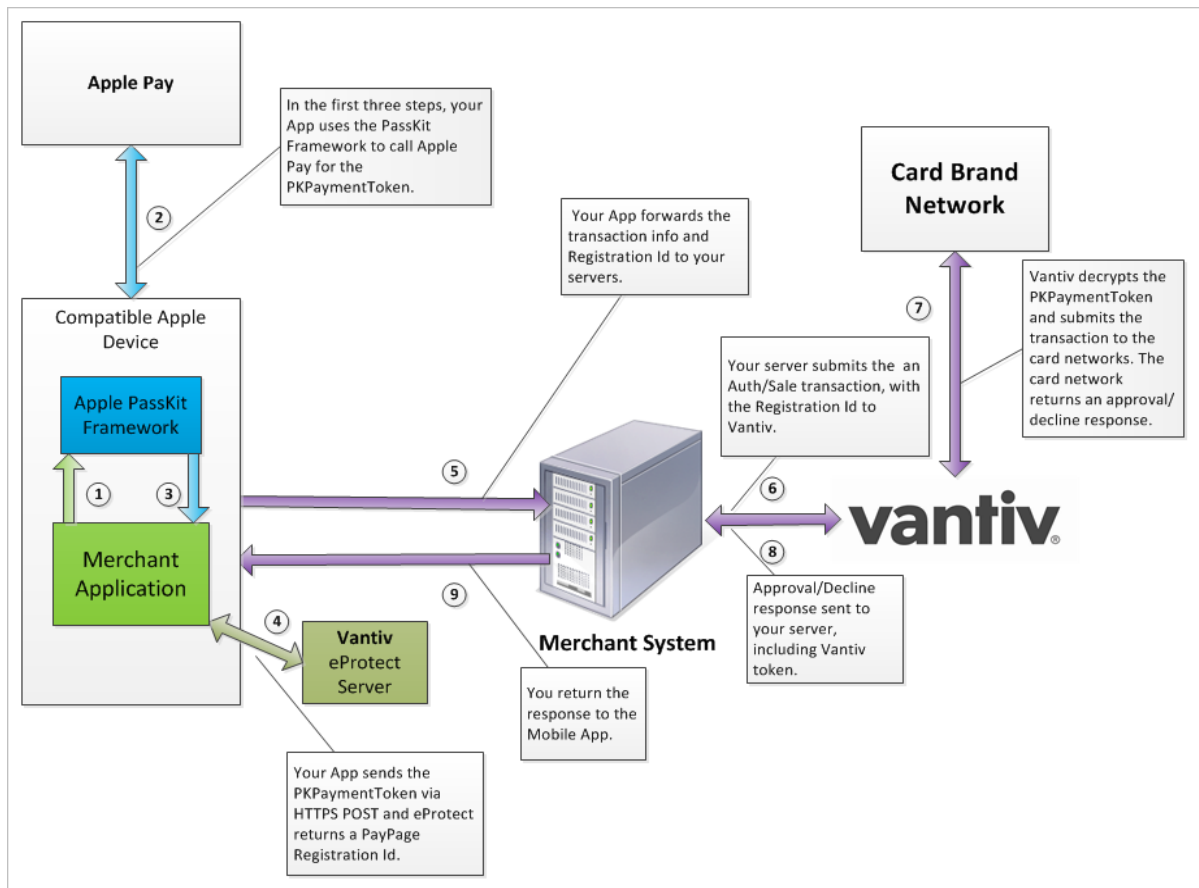
The steps that occur when a consumer initiates an Apple Pay purchase using your mobile application are detailed below and shown in [Figure 2-3](#).

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 51) 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/submit 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.

---

<b>NOTE:</b>	<b>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.</b>
--------------	--

---

**FIGURE 2-3** Data/Transaction Flow using the Vantiv Mobile API for Apple Pay



### 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 46, 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%2FTpWe10A5y9RmL5UfboTiDiVjni3zWfTty8dtv72RJL1bk%2FU4dTdlrqlT1V210TSnI%0APLdOnn
HBO51bt9Ztj9odDTQ5LD%2F4hMZTQj31BRvFOtTtjp9ysBAsydgjEjcCcbnkx7dCqgnwguz%0Ay7bX%2B5Fo8a8R
KqoprkdPwIMWOC9yWe7MQw%2FbOM5NY2QtIcIvzbLfcYUxndYtG0IXNBHNzsvUOjmw%0AvEnMhXxeCH%2BC4KoC6M
EsAGK5rH1TsdSvTzZzHF5c12dpsqdi73%2FBk6qEcdlT7gJKVmyDQC%2FNFxJ0X%0AF9930f6ejQDQj6Bzsz8X7kYC
yJdI%2FFPJZP4e3L%2FtCsBDUTJAgFLt2x8HwPaW8psILOGCCvJQm%0ATR1m70DtSChaWob7eYm1BpNiD3wkCH
8nmIMrlnt3KP4SeQ%3D%3D&applepay.signature=MIAGCSqGSIB3DQEHAqCAMIACAQExDzANBglghkgBZQMEAGE
FADCBgkqhkiG9w0BBwEAAKCAMIICvzCCAmWgAwIBAgIIQpCV6UIIb4owCgYIKoZIzj0EAwIweJumCwGA1UEAwWl
QXBwBwGUQXBwBwG1jYXRpb24gSW50ZWdyYXRpb24gQ0EgLSBHMzEmMCQGA1UECwwdQXBwBwGUQ2VydGlmawNhdGlvb
iBBdXRob3JpdHkxEzARBGNVBAoMCKFwcGx1IEluYy4xCzAJBgNVBAYTAlVTMB4XDTE0MDUwODAxMjMzOVV0XDE5MD
UwNzAxMjMzOVV0XDE5MDUwODAxMjMzOVV0XDE5MDUwODAxMjMzOVV0XDE5MDUwODAxMjMzOVV0XDE5MDUwODAxMjMzOV
5c3RlbXMeZARBGNVBAoMCKFwcGx1IEluYy4xCzAJBgNVBAYTAlVTMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE
whV37evWx7Ihj2jdcJChIY3HsL1vLCg9hGCV2Ur0pUEbg0IO2BHzQH6DMx8cVMP36zIglrrV10%2F0komJpNwPE60
B7zCB7DBFBggrBgEFBQcBAQQ5MDcWcWYKwYBBQUHAGGKWh0dHA6Ly9vY3NwLmFwcGx1LmNvbS9vY3NwMDQtYXBw
bG9vaWNhMzAxMB0GA1UdDgQWBBSUV9tv1XSbhomJdi9%2BV4UH55tYJDAMBGNVHRMBAf8EAjAAMB8GA1UdIwQYMBa
AFcPpYScRpk%2BTvJ%2BbE9ihsP6K7%2FS5LMDQGA1UdHwQtMCswKaAnoCWGI2h0dHA6Ly9jcmwYXBwBwGUy29tL2
FwcGx1YWl1jYTMuY3J5MA4GA1UdDwEB%2FwQEAWIHgDAPBgkqhkiG92NkBh0EAgUAMAAoGCCqGSM49BAMCA0gAMEUCI
QCFGdtAk%2B7wXrBV7jTwzCBLE%2B0crVL15hjiF0reLJlPGgIgXGHyYeXwrn02Zwcl5TT1W8rIqK0QuIvOn01THC
bkhVowggLMIICdaADAgECAGhJbS%2B%2F0pjalzAKBgggqhkiOPQDDAjBnMRswGQYDVQDDBJBCHBSZSBSb290IEN
BIC0grZmxJjAkBgNVBAsMHUFwcGx1IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MRMwEQYDVQKDApBCHBSZSBSBjbmMu
MQswCQYDVQGEwJVUzAeFw0xNDAlMDYyMzQ2MzBaFw0yOTA1MDYyMzQ2MzBaMHoxLjAsBgNVBAMMJUFWcGx1IEFWc
GxpY2F0aW9uIEludG9ncmF0aW9uIENBIC0grZmxJjAkBgNVBAsMHUFwcGx1IENlcnRpZmljYXRpb24gQXV0aG9yaX
R5MRMwEQYDVQKDApBCHBSZSBSBjbmMuMQswCQYDVQGEwJVUzBZBMGBYqGSM49AgEGCCqGSM49AwEHA0IABPAKEYQ
Z12SF1RpeJYEHduiAou%2Fee65N4I38S5Phm1bVZls1r1LQ13YNIk57ugj9dhf0iMt2u2Zwvsj0KYT%2FVEWjgfcw
gfQwRgYIKwYBBQUHAQEEOjA4MDYGCCSGAQUFBzABhipodHRwOi8vb2Nzc5hcHBsZS5jb20vb2Nzc5DA0LWFwcGxlc
m9vdG9hZzZwMwHQYDVRO0BBYEFcPyScRpk%2BTvJ%2BbE9ihsP6K7%2FS5LMA8GA1UdEwEB%2FwQFMAMBAf8wHwYDVR
0jBBGwFOAUu7DeoVgziJqkipnevr3rr9rLJKswNwYDVR0fBDAwLjAsCqgKIYmaHR0cDovL2Nybc5hcHBsZS5jb20
vYXBwBwGVyb290Y2FnMy5jcmwWdG9YDVR0PAQH%2FBAQDAgEGBAGCiqGSIB3Y2QGA9AgUAMAAoGCCqGSM49BAMCA2
cAMGQCMdrPcoNRFpmxhvs1w1bKYr%2F0F%2B3ZD3VNoo6%2B8ZyBxkK3ifiY95tZn5jVQQ2Pnenc%2FgIwMi3VRCG
wowV3bF3zODuQZ%2F0XfCwhbZZPxnJpghJvVPh6fRuZy5sJiSFhBpkPCZIdAAxggfMIIBWwIBATCBhjB6MS4wLA
YDVQDDCVBCHBSZSBBCHBSaWNhdGlvbiBjbnRlZ3JhdGlvbiBDQSAIECzMSYwJAYDVQQLDB1BCHBSZSBDZXJ0aWZ
pY2F0aW9uIEF1dGhvcml0eTETMBEGA1UECgwKQXBwBwGUgSW5jLjELMAkGA1UEBhMCVVMCCEKQlelCCG%2BKMA0GCW
CGSAFlAwQCAQUAoGkwGAYJKoZIhvcNAQkDMQsGCSqGSIB3DQEHAATACBgkqhkiG9w0BCQUxXcNMTQxMDA2MjE1NjQ
zWjAvBgkqhkiG9w0BCQQxIgQgg8i4X6yRAU7AXS1lamCf02UIQlpUvNPTToXUaamsFUT8wCgYIKoZIzj0EAwIERzBF
AiBe17NGTuuk%2BW901k30ac4Z90PoMhN1qRqni9KNEb%2FXAIALELZyDw0fQM8t0pXO86gg9xXFz424rEM1J0
1TM1VxhAAAAA&applepay.version=EC_v1&applepay.header.applicationData=496461ea64b50527d2
d792df7c38f301300085dd463e347453ae72debff6f4d14&applepay.header.ephemeralPublicKey=MfkwEwY
HKoZIzj0CAQYIKoZIzj0DAQcDQgAEarp8xOhLX9QliUPS9c54i3cqEfrJD37NG75ieNxcOeFLkjCk%2FBN3jVxHl
ecRwYqe%2BAWQxZBtdyewaZcmWz5lg%3D%3D&applepay.header.publicKeyHash=zoV5b2%2BmqnMiXU9avTeq
Wxc7OW3fnKXfxyhY0cyRixU%3D&applepay.header.transactionId=23e26bd8741fea9e7a4d78a69f4255b3
15d39ec14233d6f1b32223d1999fb99f"https://request.eprotect.vantivprelive.com/eProtect/payp
age
```

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": "S0ZBUURMT1ZkMTgrbW1IL3BZVFFmaDh0M0hjdDZ5RXcxQzRQUkJKRzdVc3JURXp0N0JBdmhDN05aT1lUQU5rY1RCMDhLNxg2c1I0cDV3S8k5vQmlPTjY3V2plbDVac0lqd0FkblYwVtdQWms9", "type": "VI", "id": "1234", "vantivTxnId": "82826626153431509", "message": "Success", "orderId": "PValid", "reportGroup": "*merchant1500", "response": "870", "responseTime": "2015-01-19T18:35:27", "expDate": "0718" }
```

### 2.3.3 Using the Vantiv Mobile API for Visa Checkout

The operation of Visa Checkout is simple, but requires either the modification of your existing application or development of new native applications that include the use of the Visa Checkout SDK and handling of the encrypted data returned to your application by Visa Checkout. The basic steps that occur when a consumer initiates an Visa Checkout purchase using your mobile application are:

1. When the consumer selects the Visa Checkout option from your application, your application makes use of the Visa Checkout SDK to request payment data from Visa Checkout.
2. When Visa Checkout receives the call from your application, Visa creates a Payment Success event using the Vantiv API key. Included in the Payment Success event is encrypted PAN data.
3. Visa Checkout returns the Payment Success event (defined in Visa documentation; see [https://developer.visa.com/products/visa\\_checkout/guides](https://developer.visa.com/products/visa_checkout/guides)) to your application.

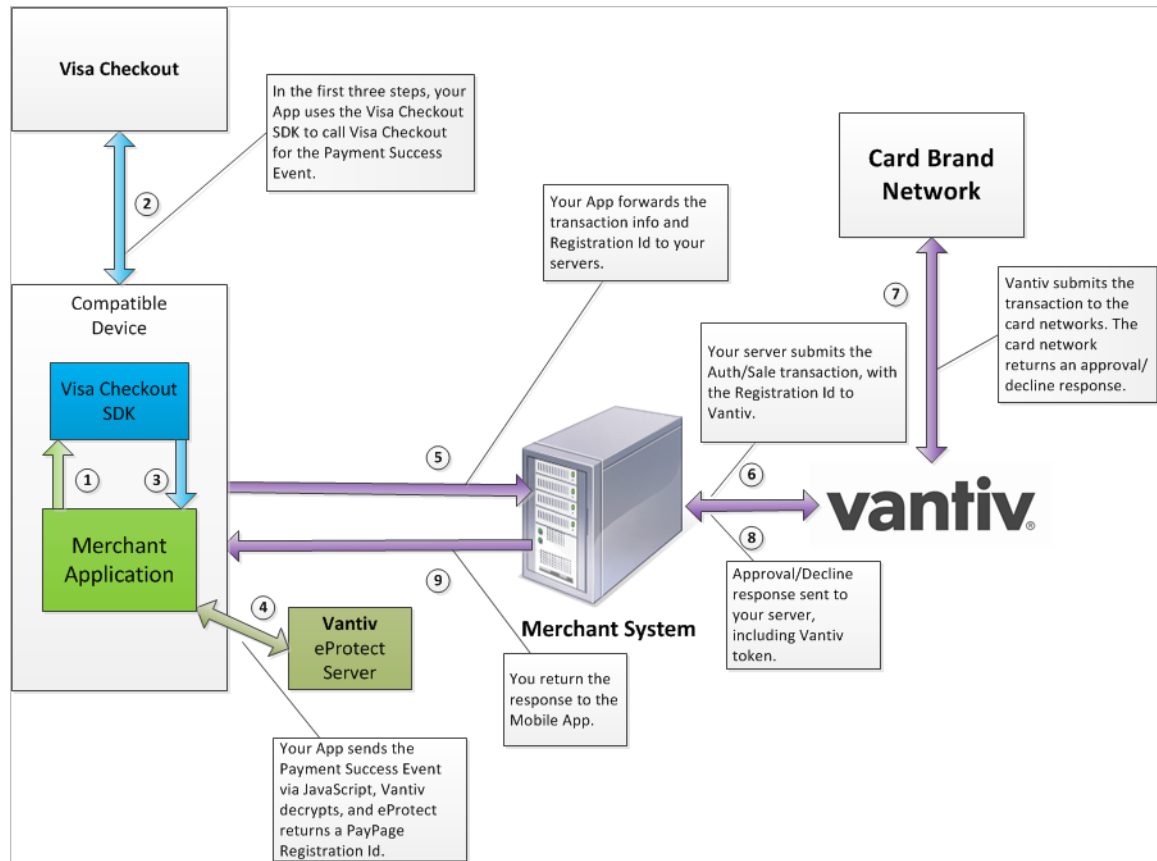
In this scenario, your native application performs an HTTPS POST of the Visa Checkout SDK using the Vantiv Mobile API for Visa Checkout. From this point forward, your handling of the transaction is identical to any other eProtect transaction. The eProtect server returns a Registration ID (low-value token) and your Mobile Application (or server) constructs the transaction using that ID (outlined in the following steps).

4. Your native mobile application sends the Payment Success event to our secure server via an HTTPS POST (see [HTTPS POST Required Components - Vantiv Mobile API for Visa Checkout](#)), Vantiv decrypts the Payment Success event associated with the Registration ID and eProtect then returns a Registration ID along with customer information from the decrypted data.
5. Your native mobile 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 Authorization/Sale transaction using the Registration ID.
7. Vantiv 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.

After you finish making a payment, you update the payment information in Visa Checkout. To update Visa Checkout from a Thank You page (next page to load after making the payment), you add a one-pixel image to the page.

**FIGURE 2-4** Data/Transaction Flow using the Vantiv Mobile API for Visa Checkout



### 2.3.3.1 Sending Vantiv the Required Fields

Construct your HTTPS POST as detailed above using the components listed in [Table 2-8](#) below. These fields take the place of the accountNumber and cvv fields from the Mobile API (encoded as form fields). See the [Sample Apple Pay POST Request](#) and [Sample Visa Checkout POST Response](#) below.

**TABLE 2-8** HTTPS POST Required Components - Vantiv Mobile API for Visa Checkout

Parameter Name	Description
visaCheckout.apiKey	The API key used to identify the shared secret used by Visa Checkout and Vantiv to decrypt the <code>encPaymentData</code> and <code>encKey</code> fields. You use both a live key and a sandbox key, which are different from each other.
visaCheckout.encKey	Encrypted key to be used to decrypt <code>encPaymentData</code> . Vantiv uses its shared secret identified by the <code>apiKey</code> to decrypt this key.
visaCheckout.encPaymentData	Encrypted consumer and payment data that can be used to process the transaction. Vantiv uses the decrypted <code>encKey</code> to decrypt this value. The decrypted value will be returned in the eProtect response.
visaCheckout.callid	Visa Checkout transaction ID associated with a payment request.

### 2.3.3.2 Sample Visa Checkout POST Request

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

```
paypageId=VgrJZt5GfhX9DYQq&reportGroup=VIReportGroup&orderId=TC7976_1_viCheckoutPPPost
&id=12345&visaCheckout.apiKey=ENIWGH9WQ4RZMHU9TPKX21EwEluZeS4zp252_mnXTRKuuvUG4&visaCh
eckout.encKey=I7Q2OFSekX8PZ239ZSX8T8AqAZwOtyzTS%2BH%2F9Rj4%2FOXQn2uDTbNIPHJY4KXu5e9UV2
Gphi3Mm2DY%2Fb947DaTU07eD0ijwIUi8wcJQZjddtoe3oC7TLVSqIvtwNb0UjXf&visaCheckout.callid=1
11111111111111111111&visaCheckout.encPaymentData=YnIJmecBIamQMxbUTVjNXJbxf%2BGarZSqvC8T%
2FWRz1AqG1jGHA3TCbu3V268uNIEDKFX4cbgHF6pxSCJjNPMrUeMef3Y3%2Fvd08GgSsucrLS4MLPwj4LBtX7G
4mtlHFii4E0ESMedNxywH%2BJ68Oqu%2BTkN%2FwHIJNfAajg2F7pHx3qDuisCih%2BPkCAWOAjrHxOS1LKf%2
FzkerBfR0U0RYZBuxsD4zWJIUfhSbtrbnmVhD%2BITZSr%2BT%2F6zvPVtInnND3yqFf%2Bmy7IymV5RUV9Ta4
1yVuEc&pciNonSensitive=false
```

### 2.3.3.3 Sample Visa Checkout POST Response

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

```
{
  "paypageRegistrationId": "1479321767965480923",
  "bin": "400552",
  "type": "VI",
  "firstSix": "400552",
  "lastFour": "4821",
  "visaCheckoutResponse": {
    "userData": {
      "userFirstName": "Jonathon",
      "userLastName": "Ross",
      "userFullName": "Jonathon Ross",
      "userName": "jross@vantiv.com",
      "encUserId": "s/XZjc5uAHsIHCrH+NwOkqfirWmMlo04lcj3TkW/3eA\\u003d\\",
      "userEmail": "jross@vantiv.com",
      "paymentRequest": {
        "merchantRequestId": "Merchant defined requestID",
        "currencyCode": "USD",
        "subtotal": "10",
        "shippingHandling": "2",
        "tax": "2",
        "discount": "1",
        "giftWrap": "2",
        "misc": "1",
        "total": "16",
        "orderId": "Merchant defined order ID",
        "description": "...corp Product",
        "promoCode": "Merchant defined promo code",
        "paymentInstrument": {
          "id": "QexjXZ5cNF/+v2nb50kXCTN2as05wC7G+gXh8iN/kaY\\u003d\\",
          "lastFourDigits": "4821",
          "binSixDigits": "400552",
          "paymentType": {
            "cardBrand": "VISA",
            "cardType": "CREDIT"
          },
          "billingAddress": {
            "personName": "Jonathon Ross",
            "firstName": "Jonathon",
            "lastName": "Ross",
            "line1": "900 Chelmsford Street",
            "line2": "Floor 11 co Vantiv eCommerce",
            "city": "Lowell",
            "stateProvinceCode": "MA",
            "postalCode": "01851",
            "countryCode": "US",
            "phone": "9782756684",
            "default": false,
            "verificationStatus": "VERIFIED",
            "expired": false,
            "cardArts": {},
            "isSuerBid": "14",
            "nameOnCard": "Jonathon Ross",
            "cardFirstName": "Jonathon",
            "cardLastName": "Ross",
            "expirationDate": {
              "month": "12",
              "year": "2020"
            },
            "shippingAddress": {
              "id": "1Y6VPBAi7ajTAS0XqKizskaC0GuTrYzYotxiC5URnDY\\u003d\\",
              "verificationStatus": "VERIFIED",
              "personName": "Jonathon Ross",
              "firstName": "Jonathon",
              "lastName": "Ross",
              "line1": "900 Chelmsford Street",
              "line2": "Floor 11 co Vantiv eCommerce",
              "city": "Lowell",
              "stateProvinceCode": "MA",
              "postalCode": "01851",
              "countryCode": "US",
              "phone": "9782756684",
              "default": false,
              "riskData": {
                "advice": "UNAVAILABLE",
                "score": "0",
                "avsResponseCode": "0",
                "cvvResponseCode": "0",
                "ageOfAccount": "1"
              },
              "newUser": false,
              "cnpTxnId": "82832924191048159",
              "orderId": "TC7976_1_viCheckoutPPPost",
              "response": "870",
              "responseTime": "2017-06-27T19:53:24",
              "message": "Success",
              "reportGroup": "VIREportGroup",
              "id": "12345"
            }
          }
        }
      }
    }
  }
}
```

### 2.3.4 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 2-5](#), 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.

---

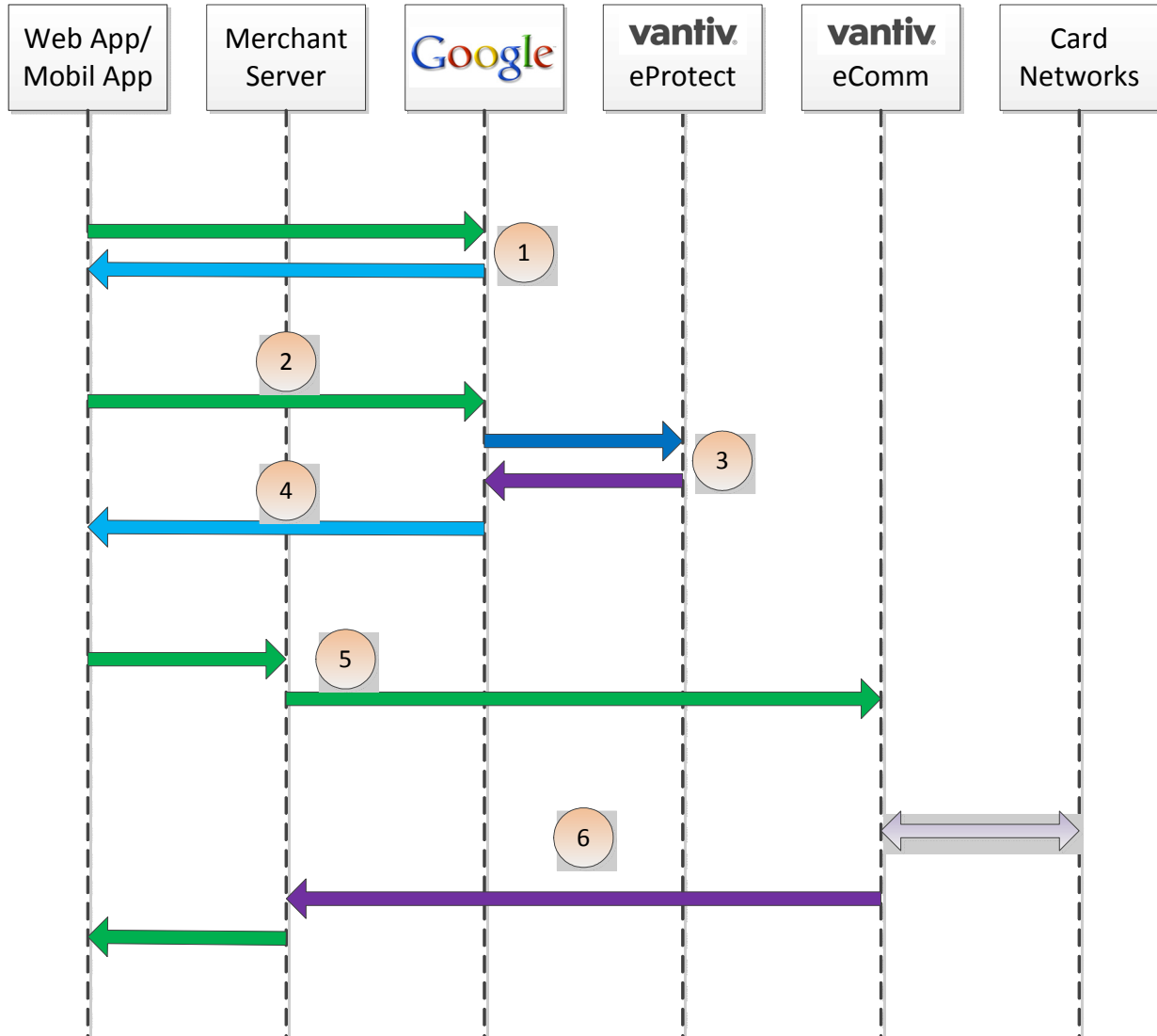
**NOTE:** Instead of submitting a Auth/Sale transaction, you can submit a Register Token transaction to convert the low-value token to a Vantiv high-value token. You would then use the high-value token in subsequent transactions submitted to the eCommerce platform.

---

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



**FIGURE 2-5** High Level Message Flow for Android Pay and Pay with Google™ using eProtect



### 2.3.5 Using the Vantiv Mobile API for Pay with Google

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 2-5](#), 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 during the testing and certification process, and provide it to your eProtect **Implementation Consultant** 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 eProtect Implementation Consultant or Vantiv Implementation Consultant if you are currently in the testing and certification process, or your Relationship Manager if you are currently in production.

## 2.5 Transaction Examples When Using ISO 8583, 610, HHMI, and PWS

This section describes how to format the Registration ID to the applicable message interface specification. These 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> returned by eProtect and the Vantiv maps the Registration ID to the OmniToken and card number.

---

**NOTE:** The Registration ID is a temporary identifier used to facilitate the mapping of a token to a card number, and 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 an unsuccessful response code based on the applicable message specification, and no token is issued.

---

Examples are included for the following message interface types on the following pages:

- [Vantiv Online Systems - Acquirer ISO 8583 Message Format](#) on page 64
- [Vantiv Online Systems - 610 Message Format](#) on page 67
- [\(HHMI\) Host-to-Host Format](#) on page 70
- [Payment Web Services \(PWS\) External Payments](#) on page 71

For further information on transaction examples with Registration ID, see the following documentation:

- *Acquirer ISO 8583 Message Format* (Effective 02.15.2017)
- *Vantiv Online Systems 610 Interface Reference Guide* (Effective 02.15.2017)
- *HHMI: Host-To-Host Format Specification* (Effective Date 02.15.2017)
- *Payment Web Services (PWS) External Payments Developers' Guide* (V6.1.2)

## 2.5.1 Vantiv Online Systems - Acquirer ISO 8583 Message Format

### Example: Request

The following example contains an authorization request with the Registration ID in field 120. If you are planning on converting a Registration ID to a network token with an Apple Pay or Android Pay cryptogram, set field 25 to '59,' (indicates *eCommerce*). Conversion to a network token is not accepted unless it is an *eCommerce* transaction.

Note that you can send the Registration ID without using 59 in field 25 (for example, if you are not using Apple Pay or Android Pay); the use of a Registration ID in general, is not specific to *eCommerce*.

```

PARSE FORMAT:  MISD          USER:  O3      DATE:  10/16/14    TIME:  14.45.35
NUM |FLDNAME|FIELD DESCRIPTION|LEN|T|FIELD VALUE
-----|-----|-----|---|---|-----
N/A |MSGTYPE|MESSAGE TYPE      |F2 |H|0200`
N/A |BITMAP1|FIRST BITMAP      |B8 |H|B238648108E080B4`
  1 |BITMAP2|SECOND BITMAP     |B8 |H|0000000000000120`
  3 |MISDPRCD|PROCESSING CODE   |F3 |H|003000`
  4 |MISDTRNA|AMOUNT, TRANSACTION|F6 |H|000000001000`
  7 |MISDTMDT|TRANSMISSION DATE AND TIME|F5 |H|`
  7 | |TRANSMISSION DATE (MMDD)|F2 |H|1015`
  7 | |TRANSMISSION TIME (HHMMSS)|F3 |H|091508`
11 |MISDSTAN|SYSTEM TRACE AUDIT NUMBER|F3 |H|091508`
12 |MISDLCTM|LOCAL TRANSACTION TIME (HHMMSS)|F3 |H|091508`
13 |MISDLCDT|LOCAL TRANSACTION DATE (MMDD)|F2 |H|1015`
18 |MISDMRHT|MERCHANT TYPE     |F2 |H|5541`
19 |MISDAQCC|ACQUIRER INST. COUNTRY CODE|F2 |H|0840`
22 |MISDPOSE|POS ENTRY MODE    |F2 |H|`
22 | |PAN/DATE ENTRY MODE      |F1 |H|01`
22 | |PIN ENTRY CAPABILITY     |F1 |H|20`
25 |MISDPOSC|POS CONDITION CODE|F1 |H|00`
32 |MISDAIID|ACQUIRING INST ID CODE|1V10|H|1042000314`
37 |MISDRRN|RETRIEVAL REFERENCE NUMBER|F12 |C|218510080021`
41 |MISDTMID|TERMINAL ID       |F15 |C|XXX12345`
42 |MISDCAID|CARD ACCEPTOR INSTITUTION ID|F15 |C|123456789`
43 |MISDTMAD|CARD ACCEPTOR NAME/LOCATION|F40 |C|`
43 | |STREET ADDRESS         |F23 |C|123 MAIN ST`
43 | |CITY                   |F13 |C|CINCINNATI`
43 | |STATE                  |F2 |C|OH`
43 | |COUNTRY                |F2 |C|US`
49 |MISDCCTR|CURRENCY CODE, TRANSACTION|F2 |H|0840`
57 | |PRODUCT TYPE           |F3 |C|REQ`
59 |MISDPGEO|NATIONAL POS GEOGRAPHIC DATA|2V14|C|2600049770`
60 |MISDARRC|ADDITIONAL POS DATA|2V36|C|`
60.1 | |TERMINAL TYPE          |F1 |C|4`
60.2 | |PHYSICAL TERMINAL LOCATION|F1 |C|1`
60.3 | |TERMINAL ENTRY CAPABILITY|F1 |C|2`
60.4 | |MERCHANT TYPE INDICATOR |F1 |C|`
60.5 | |POS CARD RETENTION INDICATOR|F1 |C|0`
60.6 | |POS TRANS STATUS INDICATOR|F1 |C|P`
60.7 | |POS TRANS ROUTING INDICATOR|F1 |C|0`
60.8 | |CHAIN NUMBER          |F6 |C|012345`
60.9 | |DIVISION NUMBER       |F3 |C|000`
60.10 | |STORE NUMBER          |F8 |C|00000123`
60.11 | |LANE NUMBER           |F3 |C|082`
60.12 | |EMPLOYEE NUMBER       |F9 |C|000000000`
62 |MISDREF|5/3 TRANSACTION DATA|2V23|C|`
62 | |5/3 TRANSACTION DATA BITMAP 1|B8 |H|4000000404004000`
62.2 | |TERMINAL SEQUENCE NUMBER|F3 |H|091508`
62.30 | |PREFERRED DEBIT ROUTING FLAG|F1 |C|0`
62.38 | |SALES TAX              |F10 |C|0000000035`
62.50 | |SALES TAX ADDENDUM FLAG|F1 |C|1`
120 |MISDADD|ADDITIONAL REQUEST DATA|2V24|C|RG0197239326028935438868`
123 |MISDMRHN|MERCHANT NAME    |F15 |C|VANTIV TEST

```

**Example: Response**

The following example contains a response with the OmniToken in field 120.

```

PARSE FORMAT:  MISD          USER:  03      DATE:  10/16/14      TIME:  14.42.39
NUM  |FLDNAME|FIELD DESCRIPTION|LEN|T|FIELD VALUE
-----|-----|-----|---|---|-----
N/A  |MSGTYPE|MESSAGE TYPE     |F2 |H|0210`
N/A  |BITMAP1|FIRST BITMAP      |B8 |H|B23A64010EF080B4`
1    |BITMAP2|SECOND BITMAP     |B8 |H|0000000000000120`
3    |MISDPRCD|PROCESSING CODE   |F3 |H|003000`
4    |MISDTRNA|AMOUNT, TRANSACTION |F6 |H|000000001000`
7    |MISDTMDT|TRANSMISSION DATE AND TIME |F5 |H|`
7    |        |TRANSMISSION DATE (MMDD) |F2 |H|1015`
7    |        |TRANSMISSION TIME (HHMMSS) |F3 |H|091508`
11   |MISDSTAN|SYSTEM TRACE AUDIT NUMBER |F3 |H|091508`
12   |MISDLCTM|LOCAL TRANSACTION TIME (HHMMSS) |F3 |H|091508`
13   |MISDLCDT|LOCAL TRANSACTION DATE (MMDD) |F2 |H|1015`
15   |MISDSTLD|SETTLEMENT DATE (MMDD) |F2 |H|1016`
18   |MISDMRHT|MERCHANT TYPE     |F2 |H|5541`
19   |MISDAQCC|ACQUIRER INST. COUNTRY CODE |F2 |H|0840`
22   |MISDPOSE|POS ENTRY MODE    |F2 |H|`
22   |        |PAN/DATE ENTRY MODE |F1 |H|01`
22   |        |PIN ENTRY CAPABILITY |F1 |H|20`
32   |MISDAIID|ACQUIRING INST ID CODE |1V10|H|1042000314`
37   |MISDLCTM|RETRIEVAL REFERENCE NUMBER |F12 |C|218510080021`
38   |MISDATID|AUTH. IDENTIFICATION RESPONSE |F6 |C|493936`
39   |MISDRSPC|RESPONSE CODE     |F2 |C|00`
41   |MISDTMID|TERMINAL ID       |F15 |C|XXX12345`
42   |MISDCAID|CARD ACCEPTOR INSTITUTION ID |F15 |C|123456789`
43   |MISDTMAD|CARD ACCEPTOR NAME/LOCATION |F40 |C|`
43   |        |STREET ADDRESS     |F23 |C|123 MAIN ST`
43   |        |CITY                |F13 |C|CINCINNATI`
43   |        |STATE               |F2 |C|OH`
43   |        |COUNTRY             |F2 |C|US`
44   |MISDRDTA|ADDITIONAL RESPONSE DATA |2V4 |C|`
44.1 |        |AUTHORIZATION SOURCE |F1 |C|`
44.2 |        |ADDRESS VERIFICATION RESULT |F1 |C|`
44.3 |        |CVV2/CVC2 RESPONSE CODE |F1 |C|N`
44.4 |        |RECURRING PAYMENT ADVICE |F1 |C|`
49   |MISDCCTR|CURRENCY CODE, TRANSACTION |F2 |H|0840`
57   |        |PRODUCT TYPE       |F3 |C|A C`
59   |MISDPGEO|NATIONAL POS GEOGRAPHIC DATA |2V14|C|2600049770`
60   |MISDARRC|ADDITIONAL POS DATA |2V7 |C|`
60.1 |        |TERMINAL TYPE      |F1 |C|4`
60.2 |        |PHYSICAL TERMINAL LOCATION |F1 |C|0`
60.3 |        |TERMINAL ENTRY CAPABILITY |F1 |C|0`
60.4 |        |MERCHANT TYPE INDICATOR |F1 |C|`
60.5 |        |POS CARD RETENTION INDICATOR |F1 |C|0`
60.6 |        |POS TRANS STATUS INDICATOR |F1 |C|0`
60.7 |        |POS TRANS ROUTING INDICATOR |F1 |C|0`
62   |MISDREF |5/3 TRANSACTION DATA |2V29|C|`
62   |        |5/3 TRANSACTION DATA BITMAP 1 |B8 |H|5E40000000000000`
62.2 |        |TERMINAL SEQUENCE NUMBER |F3 |H|091508`
62.4 |        |ACQUIRING INST ACRONYM |F4 |C|MPSM`
62.5 |        |ISSUING INST ACRONYM |F4 |C|VISN`
62.6 |        |OWNER SETTLEMENT AGENT |F4 |C|MPSM`
62.7 |        |CARDHOLDER SETTLEMENT AGENT |F4 |C|VISN`
62.10|        |POS BATCH REFERENCE NUMBER |F2 |H|0000`
120  |MISDADD|ADDITIONAL REQUEST DATA |2V24|C|TK1234567890120007`
123  |MISDMRHN|MERCHANT NAME      |F15 |C|VANTIV TEST`

```

Table 2-9 lists the Credit Response Code Mappings for ISO 8583.

**TABLE 2-9** Example Credit Response Code Mappings for ISO 8583

<b>Response Code</b>	<b>Action</b>	<b>Card Disp.</b>	<b>Description</b>
00	Approve	Return	Transaction Approved
01	Refer	Return	Refer to Card Issuer
02	Refer	Return	Refer to Card Issuer, Special Conditions
03	Decline	Return	Invalid Merchant ID
04	Decline	Keep	Pick up Card
05	Decline	Return	Generic Authorization Decline
06	Decline	Return	Error
07	Decline	Keep	Pick up Card, Special Conditions
08	Approve	Return	Honor with Identification
10	Approve	Return	Approved for Partial Amount
11	Approve	Return	VIP Approval
12	Decline	Return	Invalid Transaction
13	Decline	Return	Invalid Amount
14	Decline	Return	Invalid Account Number
15	Decline	Return	No Such Issuer
17	Decline	Return	Customer Cancellation
19	Decline	Return	Re-try Transaction
21	Decline	Return	Reversal Unsuccessful
25	Decline	Return	Unable to locate record on file
27	Decline	Return	File update field edit error
28	Decline	Return	Update file temporarily unavailable
30	Decline	Return	Message Format Error
32	Decline	Return	Partial Reversal
33	Decline	Keep	Pick Up Card - Expired
38	Decline	Keep	Allowable Number of PIN Tries Exceeded
39	Decline	Return	No Credit Amount



**TABLE 2-9** Example Credit Response Code Mappings for ISO 8583 (Continued)

Response Code	Action	Card Disp.	Description
40	Decline	Return	Requested Function Not Supported
41	Decline	Keep	Pick up Card - Lost
43	Decline	Keep	Pick up Card - Stolen
51	Decline	Return	Insufficient Funds
52	Decline	Return	No Checking Account
53	Decline	Return	No Savings Account
54	Decline	Return	Expired Card
55	Decline	Return	Incorrect PIN
56	Decline	Return	Cannot Process
57	Decline	Return	Transaction not Permitted to Cardholder
58	Decline	Return	Transaction not Permitted to Acquirer
61	Decline	Return	Exceeds Withdrawal Limit

## 2.5.2 Vantiv Online Systems - 610 Message Format

This section contains examples of an Authorization with a Registration ID, a Sale with a Registration ID request, and a Token Conversion Request using the 610 Interface format (where \* = space, <rs> = record separator, <gs>= group separator).

### Example: Authorization Request

Example of Credit Card Authorization Request using Registration-ID:

```
|I2.|123456|0100|21|004000|123456789|123456|1234|123456|123456|123456|011|0000000600|11
11|222|333333333333|123|*****
*****|12345678|12345678|123456789|123|12|PO#/CUSTOMER*CODE***|123456789|TRACE*DATA*1*
**|<rs>G028|1234567890123456789|R*****|1234|<gs>
```

### Example: Registration ID Request through Financial Transaction (sale)

Example of Credit Card Sale Request using Registration-ID:

```
|I2.|123456|0200|22|004000|000001500|0321031116|123456|032103|111600|812|0000000600|1111|2
22|333333333333|001|*****
*****|12345678|00000001|000000000|00|000|40|PO#/CUSTOMER*CODE***|000000000|TRACE*DATA*1*
**|<rs>G028|1234567890123456789|R*****|1234|<gs>
```

### Example: Token Conversion Request

Example of Token Conversion Request for a low-value token:

```
|I2.|123456|0100|50|800000|0321031116|123456|032103|111600|812|0000000600|1111|222|3333333
33333|001|*****|
45678|00001234|000|00|TRACE*DATA*1***|99999999|999999|
```

For more information, see Appendix B, “Special Transaction Provisioning,” in the *Vantiv Online Systems 610 Interface Reference Guide* (Effective 02.15.2017).

## 2.5.2.1 Response

### Example: Token Approval Response

```
|0110|53|123456|0321031116|123456|TRACE*DATA*1***|138001|N|
```

If host tokenization fails (F value) when processing Legacy or Omni tokens, the token contains spaces and the Token ID contains ZZZZZZ. The host attempts to process transaction requests without token data (using clear PAN or Track) when applicable. The transaction may result in an approval or decline using E2EE and/or clear PAN/track data.

If host tokenization fails when processing the Registration ID, the token field and token-id field is not returned in R017. The error is indicated by presence of this Field Type 4, Value F. The transaction results in a decline if a token can not be created using the Registration ID.

Table 2-10 lists some of the most common TPS codes, including *416 - REGISTRATION ID NOT FOUND*. These codes are not necessarily tied to a specific network or product. Vantiv returns them for any card type depending on the transaction disposition.

**TABLE 2-10** TPS Global Response Codes

TPS	Response Message	Description
001	AUTH DOWN	This is the TPS response when no external authorizer is available.
307	CONVERSION TRAN ERR	<b>Description:</b> Token/De-token conversion transaction not allowed <b>Action:</b> Check Host tokenization configuration settings and request message
307	PAYMENT TYPE ERROR	<b>Description:</b> Host payment type setup configured to decline token transaction request <b>Action:</b> Check Host tokenization configuration settings and request message
307	ADDITIONAL TRAN ERR	<b>Description:</b> Transaction not allowed to use token <b>Action:</b> Check Host tokenization configuration settings and request message

**TABLE 2-10** TPS Global Response Codes (Continued)

TPS	Response Message	Description
307	RESTRICT TRAN ERROR	<b>Description:</b> Restricted transactions not allowed to process <b>Action:</b> Check Host tokenization configuration settings and request message
307	INVALID TRACK DATA	<b>Description:</b> Track data is blank, unable to tokenize transaction request <b>Action:</b> Verify values for clear PAN or Track data - retry transaction
307	INV POS COND CODE	<b>Description:</b> POS condition code is invalid for request message <b>Action:</b> Verify values for POS condition code field - retry transaction
307	INV ORIG DATE TIME	<b>Description:</b> Invalid token original date or time values, not numeric or format issue <b>Action:</b> Verify POS values for token original date and time fields - retry transaction
307	TRAN NOT ALLOWED	<b>Description:</b> Transaction requesting or using token is not allowed <b>Action:</b> Verify POS request message - retry transaction
307	POS ENTRY MODE ERROR	<b>Description:</b> Transaction request using token must set manual POS entry mode <b>Action:</b> Verify POS request message - retry transaction
307	PAN OR TRACK ERROR	<b>Description:</b> Transaction using token must initialize track and/or PAN to spaces <b>Action:</b> Verify POS request message - retry transaction
416	REGISTRATION -ID NOT FOUND	<b>Description:</b> The Registration-ID used in the transaction could not be found in the ES data base. It may have expired. <b>Action:</b> Check the Registration-ID and retry the transaction.
714	INV CARD NUMBER	This is the TPS response code that will most commonly be presented when a transaction fails to locate a viable routing option.
796	AUTH DOWN	System malfunction

For more information and a complete list, see Appendix A, “TPS Response Codes,” in the *Vantiv Online Systems 610 Interface Reference Guide* (Effective 02.15.2017).

## 2.5.3 (HHMI) Host-to-Host Format

The following example contains an authorization request with the Registration ID in field 25, using HOST-to-HOST Format.

### Example: Request

```

PARSE FORMAT: HHMI          USER: O3          DATE: 10/16/14    TIME: 14.25.51
NUM |FLDNAME|FIELD DESCRIPTION|LEN|T|FIELD VALUE
-----|-----|-----|----|---|-----
N/A |1|MESSAGE ID|F4|C|EFQC`
N/A |2|MESSAGE SEQUENCE NUMBER|F6|C|164259`
N/A |3|TRANSACTION DATE (YYMMDD)|F6|C|141015`
N/A |4|TRANSACTION TIME (HHMMSS)|F6|C|164259`
N/A |5|TERMINAL OWNER ID|F4|C|MPSM`
N/A |9|TERMINAL OWNER|F10|C|922222222`
N/A |10|TERMINAL TYPE|F2|C|03`
N/A |12|TERMINAL NUMBER (RESP ONLY)|F6|C|`
N/A |13|TERMINAL SEQUENCE NUMBER|F6|C|164259`
N/A |15|SETTLEMENT DATE (RESP ONLY)|F6|C|`
N/A |16|TRANSMISSION DATE (YYMMDD)|F6|C|141015`
N/A |17|SHARING GROUP|F6|C|9E80EA`
N/A |18|REGIONAL MAP NUMBER|F2|C|18`
N/A |19|CARDHOLDER ID|F4|C|SWTH`
N/A |21|CARDHOLDER R&T NUM (RESP ONLY)|F10|C|`
N/A |22|CARDHOLDER NTWK ID (RESP ONLY)|F4|C|`
N/A |23|RESERVED|F10|C|`
N/A |23|RESERVED|F10|C|`
N/A |24|TRANSACTION TYPE|F2|C|46`
N/A |25|TRANSACTION AMOUNT|F10|C|0000000100`
N/A |26|FROM ACCOUNT|F3|C|CC`
N/A |27|TO ACCOUNT|F3|C|`
N/A |28|TRAN PROGRESS IND / PATH ID|F2|C|01`
N/A | |VARIABLE ADDENDUM FIELDS|D2|C|`
NA.04|31|ACQUIRER TERMINAL IDENT.|3V8|C|XXX12345`
NA.05|32|TERMINAL LOCATION LENGTH|F3|C|044`
NA.05|32|TERMINAL STREET ADDR|F27|C|123 MAIN ST`
NA.05|32|TERMINAL CITY|F15|C|CINCINNATI`
NA.05|32|TERMINAL STATE|F2|C|OH`
NA.07|34|TERMINAL BUSINESS DATE (YYMMDD)|3V6|C|141015`
NA.09|99|SALES TAX ADDENDUM FLAG|3V1|C|0`
NA.25|126|E2E / TOKEN DATA|3V24|C|RG0197239326028935438868`
NA.41|44|MERCHANT ACCOUNT NUMBER|3V20|C|123456789`
NA.42|45|POS MERCHANT DATA|3V20|C|840010000005411`
NA.47|47|MERCHANT REPORTING IDENT.|3V24|C|800200000001008000000000`
NA.99|56|END SENTINEL|F0|C|`

```

The following example contains an authorization response in field 25, using HOST-to-HOST Format.

**Example: Response**

```

PARSE FORMAT: HHMI          USER: 03          DATE: 10/16/14    TIME: 14.34.40
NUM |FLDNAME |FIELD DESCRIPTION          |LEN |T|FIELD VALUE
-----|-----|-----|-----|-----|-----
N/A |1|MESSAGE ID                  |F4 |C|EFRC`
N/A |2|MESSAGE SEQUENCE NUMBER    |F6 |C|164259`
N/A |3|TRANSACTION DATE (YYMMDD)  |F6 |C|141015`
N/A |4|TRANSACTION TIME (HHMMSS)  |F6 |C|164259`
N/A |5|TERMINAL OWNER ID          |F4 |C|MPSM`
N/A |9|TERMINAL OWNER              |F10 |C|922222222`
N/A |10|TERMINAL TYPE               |F2 |C|03`
N/A |12|TERMINAL NUMBER (RESP ONLY)|F6 |C|016731`
N/A |13|TERMINAL SEQUENCE NUMBER    |F6 |C|164259`
N/A |15|SETTLEMENT DATE (RESP ONLY)|F6 |C|141015`
N/A |16|TRANSMISSION DATE (YYMMDD)  |F6 |C|141015`
N/A |17|SHARING GROUP              |F6 |C|9E80EA`
N/A |18|REGIONAL MAP NUMBER        |F2 |C|18`
N/A |19|CARDHOLDER ID              |F4 |C|VISM`
N/A |21|CARDHOLDER R&T NUM (RESP ONLY)|F10 |C|1000000000`
N/A |22|CARDHOLDER NTWK ID (RESP ONLY)|F4 |C|VISM`
N/A |23|RESERVED                    |F10 |C|`
N/A |24|TRANSACTION TYPE           |F2 |C|46`
N/A |25|TRANSACTION AMOUNT        |F10 |C|0000000100`
N/A |26|FROM ACCOUNT                |F3 |C|CC`
N/A |27|TO ACCOUNT                 |F3 |C|`
N/A |28|TRAN PROGRESS IND / PATH ID |F2 |C|01`
N/A | |VARIABLE ADDENDUM FIELDS    |D2 |C|`
NA.04|31|ACQUIRER TERMINAL IDENT.      |3V8 |C|XXX12345`
NA.06|33|RESPONSE ADVICE             |3V5 |C|0R000`
NA.08|35|PROCESSOR BUSINESS DATE      |3V6 |C|141015`
NA.12|37|FROM ACCOUNT                |3V22 |C|CC`
NA.25|126|E2E / TOKEN DATA             |3V24 |C|TK1234567890120007
NA.40|43|AUTHORIZATION STATUS           |3V8 |C|AA241586`
NA.41|44|MERCHANT ACCOUNT NUMBER        |3V20 |C|123456789
NA.47|47|MERCHANT REPORTING IDENT.        |3V15 |C|800200000001008`
NA.63|53|POS BATCH REFERENCE NUMBER      |3V4 |C|0000`
NA.85|85|CVV2                      |3V1 |C|N`
NA.99|56|END SENTINEL                    |F0 |C|`

```

**2.5.4 Payment Web Services (PWS) External Payments**


---

**NOTE:** eProtect in combination with PWS is available to existing PWS customers only.

---

The example below contains an authorization request with Registration ID.

For response codes, please see the “Response” section of the *Payment Web Services (PWS) External Payments Developers' Guide - V6.1.2*. For unsuccessful attempts, see the Troubleshooting section for SOAP Faults List.

**Example: Authorization Request**

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:urn="urn:com:vantiv:types:payment:transactions:v1"
  xmlns:urn1="urn:com:vantiv:types:payment:systems:v1"
  xmlns:urn2="urn:com:vantiv:types:common:v1"
  xmlns:urn3="urn:com:vantiv:types:payment:instruments:v1"
  xmlns:urn4="urn:com:vantiv:types:payment:transactions:v6"
  xmlns:urn5="urn:com:vantiv:types:payment:systems:v6"
  xmlns:urn6="urn:com:vantiv:types:common:v6"
  xmlns:urn7="urn:com:vantiv:types:payment:instruments:v6">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-w
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utilit
        <wsse:UsernameToken wsu:Id="UsernameToken-4">
          <wsse:Username></wsse:Username>
          <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-u
        </wsse:UsernameToken>
      </wsse:Security>
    </soapenv:Header>
    <soapenv:Body>
      <urn4:AuthorizeRequest system-trace-id="123456" merchant-ref-id="1A">
        <urn4:Merchant>
          <urn4:MerchantId></urn4:MerchantId>
          <urn4:NetworkRouting></urn4:NetworkRouting>
          <urn4:MerchantName></urn4:MerchantName>
          <urn4:ClerkNumber></urn4:ClerkNumber>
          <!--Optional:-->
          <urn4:CashierNumber></urn4:CashierNumber>
          <!--Optional:-->
          <urn4:LaneNumber></urn4:LaneNumber>
          <!--You have a CHOICE of the next 3 items at this level-->
          <urn4:DivisionNumber></urn4:DivisionNumber>
          <urn4:ChainCode></urn4:ChainCode>
          <urn4:StoreNumber></urn4:StoreNumber>
          <urn5:Terminal>
            <!--You have a CHOICE of the next 2 items at this level-->
            <urn6:IPv4Address></urn6:IPv4Address>
            <urn5:TerminalID></urn5:TerminalID>
            <!--Optional:-->
            <urn5:SequenceNumber></urn5:SequenceNumber>
            <urn5:Classification></urn5:Classification>
            <!--Optional:-->
            <urn5:EntryMode></urn5:EntryMode>
            <!--Optional:-->
            <urn5:CardReader></urn5:CardReader>
            <!--Optional:-->
            <urn5:PinEntry></urn5:PinEntry>
            <!--Optional:-->
            <urn5:BalanceInquiry></urn5:BalanceInquiry>
          </urn5:Terminal>
        </urn4:Merchant>
        <urn4:TransactionType></urn4:TransactionType>
        <urn4:TransactionTimestamp></urn4:TransactionTimestamp>
        <urn4:PaymentType></urn4:PaymentType>
        <!--Optional:-->
        <urn4:DraftLocatorId></urn4:DraftLocatorId>
        <urn4:TokenRequested></urn4:TokenRequested>
        <urn4:TransactionAmount currency="USD"></urn4:TransactionAmount>
        <!--You have a CHOICE of the next 2 items at this level-->

```

```
<urn4:Credit>
  <urn7:CardholderAddress>
    <!--1 to 5 repetitions:-->
    <urn6:AddressLine></urn6:AddressLine>
    <urn6:City></urn6:City>
    <urn6:State></urn6:State>
    <urn6:PostalCode></urn6:PostalCode>
    <urn6:CountryCode></urn6:CountryCode>
  </urn7:CardholderAddress>
  <!--Optional:-->
  <urn7:PartialApprovalCode></urn7:PartialApprovalCode>
  <urn7:CardType></urn7:CardType>
  <!--You have a CHOICE of the next 2 items at this level-->
  <urn7:CardKeyed>
    <urn7:RegistrationId></urn7:RegistrationId>
    <urn7:ExpirationDate></urn7:ExpirationDate>
  </urn7:CardKeyed>
</urn4:Credit>
</urn4:AuthorizeRequest>
</soapenv:Body>
</soapenv:Envelope>
```

The example below contains an authorization response.

### Example: Authorization Response

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ns5:AuthorizeResponse merchant-ref-id="1A*" system-trace-id="123456" xmlns:ns7="urn:com:vantiv:types:
xmlns:ns6="urn:com:vantiv:types:common:v6" xmlns:ns5="urn:com:vantiv:types:payment:transactions:v6">
      <ns5:RequestId/></ns5:RequestId>
      <ns5:DemoMode/></ns5:DemoMode>
      <ns5:ReferenceNumber/></ns5:ReferenceNumber>
      <ns5:TransactionStatus/></ns5:TransactionStatus>
      <ns5:TransmissionTimestamp/></ns5:TransmissionTimestamp>
      <ns5:TransactionTimestamp/></ns5:TransactionTimestamp>
      <ns5:JulianDay/></ns5:JulianDay>
      <ns5:BatchNumber/></ns5:BatchNumber>
      <ns5:CardCategory/></ns5:CardCategory>
      <ns5:AuthorizationCode/></ns5:AuthorizationCode>
      <ns5:AddressVerificationResult>
        <ns5:Code/></ns5:Code>
        <ns5:Type/></ns5:Type>
      </ns5:AddressVerificationResult>
      <ns5:CardSecurityCodeResult>
        <ns5:Code/></ns5:Code>
        <ns5:Type/></ns5:Type>
      </ns5:CardSecurityCodeResult>
      <ns5:PaymentServiceResults>
        <ns5:VisaResults>
          <ns5:TransactionId/></ns5:TransactionId>
          <ns5:ValidationCode/></ns5:ValidationCode>
          <ns5:AuthorizationCharacteristicsIndicator/></ns5:AuthorizationCharacteristicsIndicator>
          <ns5:CardLevelResultsCode/></ns5:CardLevelResultsCode>
        </ns5:VisaResults>
      </ns5:PaymentServiceResults>
      <ns5:TokenizationResult>
        <ns5:successful/></ns5:successful>
        <ns5:tokenType>
          <ns7:tokenValue/></ns7:tokenValue>
        </ns5:tokenType>
      </ns5:TokenizationResult>
      <ns5:NetworkResponseCode/></ns5:NetworkResponseCode>
    </ns5:AuthorizeResponse>
  </soapenv:Body>
</soapenv:Envelope>
```



## 2.6 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 108 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.

### 2.6.1 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.

## 2.6.2 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.

---

### 2.6.2.1 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>Registration 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>

```

### 2.6.2.2 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>
```

## 2.6.3 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.

---

### 2.6.3.1 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>Registration 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.vantivcnpi.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>

```

### 2.6.3.2 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>
```

## 2.6.4 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.

### 2.6.4.1 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
```



```

9KK3p1L1p1VzE4ZWVPQV1SUHNITG1JN2I0NzlyTg=</paypageRegistrationId>
</registerTokenRequest>
</cnpOnlineRequest>

```

### 2.6.4.2 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>

```

## 2.6.5 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.

---

### 2.6.5.1 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>Registration 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>

```

### 2.6.5.2 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>

```

## 2.6.6 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 (Card Number, Order Id, Merchant Id, Amount, Approval Code, and (Auth) Response 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.

---

### 2.6.6.1 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>Registration 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>
```

### 2.6.6.2 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>
```

## 2.6.7 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.

---

### 2.6.7.1 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>Registration 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.vantivcnpi.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>

```

### 2.6.7.2 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>
```

## 2.7 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 your eProtect Implementation Consultant and Vantiv Conversion Manager.

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 Environment](#) on page 12 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 native applications:**

- Request and receive Registration ID from eProtect.
- Submit Registration ID to Vantiv for authorization (or non-financial) request for OmniToken and response.

**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 14 for definitions of the response codes.

## 2.7.1 Testing eProtect Transactions

To request and receive a Registration ID from eProtect:

1. Verify that your checkout page or mobile native application is coded correctly. See one of the following sections for more information:
  - [Integrating Customer Browser JavaScript API Into Your Checkout Page](#) on page 24.
  - [Integrating iFrame into your Checkout Page](#) on page 38.
  - [Integrating eProtect Into Your Mobile Application](#) on page 46.
2. Verify that you are using the appropriate URL (see [Table 1-2, "eProtect Certification, Testing, and Production URLs"](#) on [page 13](#)) for the testing and certification environment, for example:

```
https://request.eprotect.vantivprelive.com/eProtect/eProtect-api3.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 2-11](#). When performing these tests, you can use any expiration date and card type.
4. Verify that your results match the **Result** column in [Table 2-11](#).

**TABLE 2-11** Expected eProtect Test Results

Test Case	Card Number	Card Validation Number	Response Code	Result
<b>NOTE:</b> Card Numbers are split into two parts; join <b>Part 1</b> and <b>Part 2</b> 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)."  <b>Note:</b> Do not use when testing iFrame.

**TABLE 2-11** Expected eProtect Test Results (Continued)

Test Case	Card Number	Card Validation Number	Response Code	Result
4	Part 1: 601101 Part 2: 000003	Any 3-digit	872	Checkout form displays error message to card holder, for example, "Invalid Card Number - Check and retry (too short)."
5	Part 1: 44570100 Part 2: B00000006	Any 3-digit	874	Checkout form displays error message to card holder, for example, "Invalid Card Number - Check and retry (not a number)."
6	Part 1: 60110100 Part 2: 00000003	Any 3-digit	875	Checkout form displays error message to card holder, for example, "We are experiencing technical difficulties. Please try again later or call 555-555-1212."
7	Part 1: 51234567 Part 2: 898010003	Any 3-digit	876	Checkout form displays error message to card holder, for example, "Invalid Card Number - Check and retry (failure from server)."
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)".

**TABLE 2-11** Expected eProtect Test Results (Continued)

Test Case	Card Number	Card Validation Number	Response Code	Result
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)".  <b>Note:</b> 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 76).
2. Submit three Authorization transactions using the eProtect data from [Table 2-11](#).
3. Verify that your `authorizationResponse` values match the **Response Code** column.

To test the submission of the Registration ID to Vantiv for authorization, or a non-financial request for an OmniToken and the response:

1. Verify that your applicable message specification template is coded correctly for this transaction type.
2. Submit transactions using the eProtect Registration ID.

Verify that your response values match the expected results provided by your Vantiv Conversion Manager.



---

## 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></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 eProtect Implementation Consultant for the appropriate production URL).

---

```
<HTML>
<head>
<title>eProtect 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-api3.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,
                "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></td></tr>
          <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>

        <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-api3 */
  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-client3.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></td></tr>
```

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>Response Time</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></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></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>
        <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-client3.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 XML authorization is submitted with the sensitive card data after your customer completes the checkout form, and a token is stored in its place. The following is an example of the information sent to your order handling system:

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

### A.2.2 Information Sent with Browser-Based 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 Application 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.

```
cvv - 123  
accountNum - <account number here>  
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, as well as XML examples, see [Transaction Examples When Using cnpAPI](#) on page 75. For a comprehensive list of all cnpAPI elements and usage, see Chapter 4, “cnpAPI Elements” in the *Vantiv cnpAPI Reference Guide*.

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

- [cardValidationNum](#)
- [checkoutId](#)
- [expDate](#)
- [paypage](#)
- [paypageRegistrationId](#)
- [registerTokenRequest](#)
- [registerTokenResponse](#)
- [token](#)

### A.3.1 cardValidationNum

The `<cardValidationNum>` element is an optional child of the `<card>`, `<paypage>`, `<token>`, `<registerTokenRequest>`, or `<updateCardValidationNumOnToken>` 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 checkoutId

The `checkoutId` element is an optional child of the `token` element specifying the low-value token replacing the CVV value. You use this feature when you already have the consumer's card (i.e., token) on file, but wish the consumer to supply the CVV value for a new transaction. This LVT remains valid for 24 hours from the time of issue.

**Type** = String; **minLength** = 18; **maxLength** = 18

**Parent Elements:**

[token](#)

**Attributes:**

None

**Child Elements:**

None

### A.3.3 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.3.4 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.3.5 **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.3.6 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. <b>minLength</b> = N/A <b>maxLength</b> = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. <b>minLength</b> = N/A <b>maxLength</b> = 50
reportGroup	String	Yes	Required attribute defining the merchant sub-group in eCommerce iQ where this transaction displays. <b>minLength</b> = 1 <b>maxLength</b> = 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.3.7 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 <code>registerTokenRequest</code> transaction. <b>minLength</b> = N/A <b>maxLength</b> = 25
customerId	String	No	The response returns the same value submitted in the <code>registerTokenRequest</code> transaction. <b>minLength</b> = N/A <b>maxLength</b> = 50
reportGroup	String	Yes	The response returns the same value submitted in the <code>registerTokenRequest</code> transaction. <b>minLength</b> = 1 <b>maxLength</b> = 25

### Child Elements:

Required: [cnpTxnId](#), [response](#), [message](#), [responseTime](#)

Optional: [eCheckAccountSuffix](#), [cnpToken](#), [bin](#), [type](#), [applepayResponse](#), [androidpayResponse](#)

### A.3.8 token

The `token` element replaces the `card` element in tokenized card transactions or the `echeck` element in eCheck transactions, and defines the tokenized payment card/account information.

#### Parent Elements:

[authorization](#), [captureGivenAuth](#), [credit](#), [forceCapture](#), [sale](#), [accountUpdate](#), [updateSubscription](#)

#### Attributes:

None

---

**IMPORTANT:** Although not a required element, Vantiv recommends you include the `expDate` element. If you converted PAN information to tokens using the `registerTokenRequest` transaction, we do not have the `expDate` value stored, so cannot add it to the transaction. Transactions without `expDate` have a high likelihood of decline.

---

#### Child Elements:

Required: [cnpToken](#)

Optional: [expDate](#), [cardValidationNum](#), [type](#), [checkoutId](#)

#### Example: token Structure with CVV

```
<token>
  <cnpToken>Token</cnpToken>
  <expDate>Card Expiration Date</expDate>
  <cardValidationNum>Card Validation Number</cardValidationNum>
  <type>Method of Payment</type>
</token>
```

#### Example: token Structure with checkoutId instead of CVV

```
<token>
  <cnpToken>Token</cnpToken>
  <expDate>Card Expiration Date</expDate>
  <type>Method of Payment</type>
  <checkoutId>Low Value Token for CVV</checkoutId>
</token>
```

---

## 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 16 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-24, "CSS Properties Excluded From the White List \(not allowed\)"](#) for more information.

---

<b>NOTE:</b>	<b>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>
--------------	--

---

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

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

**TABLE B-2** Background and Border Properties (Continued)

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

**TABLE B-2** Background and Border Properties (Continued)

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

**TABLE B-3** Basic Box Properties

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

**TABLE B-3** Basic Box Properties (Continued)

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

**TABLE B-4** Flexible Box Layout

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

**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
<b>@font-face (Do not use)</b>	<i>A rule that allows websites to download and use fonts other than the “web-safe” fonts</i>

**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 <u>direction</u> 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
<i>list-style-image</i> <b>(Do not use)</b>	<i>Specifies an image as the list-item marker</i>
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
<i>cursor</i> <b>(Do not use)</b>	<i>Specifies the type of cursor to be displayed</i>
<i>icon</i> <b>(Do not use)</b>	<i>Provides the author the ability to style an element with an iconic equivalent</i>
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

**TABLE B-23** Appearance Properties

Property	Description
webkit-appearance	Used by WebKit-based (e.g., Safari) and Blink-based (e.g., Chrome, Opera) browsers to display an element using platform-native styling based on the operating system's theme.
moz-appearance	Used in Firefox to display an element using platform-native styling based on the operating system's theme.
appearance	Allows you to make an element look like a standard user interface element.

## B.2 Properties Excluded From White List

Table B-24 lists the CSS Properties that are not permitted for use when building a CSS for eProtect iFrame.

**TABLE B-24** 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

---

## 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 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?**  
*If the response from the primary server takes more than five (5) seconds, the request is automatically sent to our secondary server. To ensure the secondary server has time to respond, we recommend a timeout value of 15000 (15 seconds).*

\_\_\_\_\_ 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

---

## A

---

Android Pay, 9, 56  
Apple Pay  
    data/transaction flow, 50, 54  
    using mobile API, 48  
Apple Pay Web, 8  
    compatible devices, 9  
    data/transaction flow, customer Browser  
        JavaScript API, 34  
Authorizations  
    request structure, 76  
    response structure, 77  
Availability of the PayPage API  
    detecting, 32

## C

---

Callbacks  
    failure, 31  
    handling, 30  
    success, 30  
    timeout, 32  
Capture Given Auth Transactions, 86  
    request structure, 86  
    response structure, 88  
Checkout Form Submission  
    intercepting, 29  
cnpAPI Elements  
    checkoutId, 110  
    expDate, 111  
    paypage, 109  
    paypageRegistrationId, 113  
    registerTokenRequest, 114  
    registerTokenResponse, 115  
    token, 116  
Collecting Diagnostic Information, 62  
Contact Information, xii  
Credit Transactions, 89  
    request structure, 89  
    response structure, 90  
CSS Properties for iFrame API, 117

## D

---

Diagnostic Information  
    collecting, 62  
Document Structure, xi  
Documentation Set, xi  
Duplicate Detection, 15

## E

---

eProtect  
    Getting Started, 6  
    How it works, 4  
    Overview, 2  
expDate, 111

## F

---

Force Capture Transactions, 84  
    request structure, 84  
    response structure, 85

## H

---

HTML Checkout Page Examples, 98  
    iFrame-Integrated Checkout Page, 102  
    JavaScript API-Integrated Checkout page, 99  
    Non-eProtect Checkout Page, 98

## I

---

iFrame API, 2  
    integrating into your checkout page, 38  
Integration Steps, 24  
Intended Audience, v

## J

---

JavaScript Customer Browser API, 2  
jQuery Version, 12

## L

---

Loading the PayPage API, 25

---

**M**

---

Migrating From Previous Versions, 6  
    from JavaScript Browser API to iFrame, 7  
    from PayPage with jQuery 1.4.2, 6  
Mobile API, 2, 46  
    Information Sent, 107  
Mobile Application  
    Integrating eProtect, 46  
Mobile Operating System Compatibility, 8  
Mouse Click  
    handling, 27

---

**N**

---

Non-eProtect Checkout Page, 98

---

**O**

---

Online Authorization Request, 76  
Online Authorization Response, 78  
Online Sale Request, 79  
Online Sale Response, 81  
Order Handling Systems  
    Information Sent, 106

---

**P**

---

PayPage API Request Fields  
    specifying, 26  
PayPage API Response Fields  
    specifying, 27  
paypageRegistrationId, 113  
PCI Non-Sensitive Value Feature, 43  
POST  
    Sample Response, 48  
POST Request, 46

---

**R**

---

Register Token Transactions, 82  
    request structure, 82  
    response structure, 83  
registerTokenRequest, 114  
registerTokenResponse, 115  
Registration ID Duplicate Detection, 15  
Response Codes, 14  
Revision History, v

---

**S**

---

Sale Transactions, 79  
    request structure, 79  
    response structure, 80  
Sample JavaScripts, 12

---

**T**

---

Testing and Certification, 63  
Testing PayPage Transactions, 93  
token, 116  
TPS Global Response Codes, 68  
Transactions  
    authorization, 76  
    capture given auth, 86  
    credit, 89  
    force Capture, 84  
    register token, 82  
    sale, 79  
    types, 75

---

**V**

---

Visa Checkout  
    eProtect Support, 10  
    getting started, 10  
    requirements for use, 11  
    using the Customer Browser JavaScript  
        API, 35