worldpay

eProtect Integration Guide

December 2022

eProtect API V3.0

Document Version 4.19

Worldpay eProtect Integration Guide 4.19

CONTENTS

About This Guide

Chapter 1 Introduction

Chapter 2 Integration and Testing

Appendix A Code Samples and Other Information

worldpay

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 Worldpay.

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)
4.19	Added a note for using High Availability with the Mobile API POST option.	Chapter 1
	Changed additional instances of the maximum field length value for `orderId` from 256 to 25.	Chapter 2
	Removed Note on alternative to submitting an Auth/Sale transaction (and submitting a Register Token transaction) as it no longer applies.	Chapter 2
4.18	Added information on new optional `checkoutWithEnter` property for capturing the Enter key message event from the iFrame.	Chapter 2, Appendix A
4.17	Added two new optional properties for configuring a custom iFrame title and custom labels without using the CSS: `iFrameTitle` and `label`.	Chapter 2, Appendix A
	Changed an incorrect maximum field length value for `orderId` from 256 to 25.	

TABLE 1 Document Revision History (Continued)

Doc. Version	Description	Location(s)
4.16	Updated Sample Apple Pay POST Request example text with missing spaces.	Chapter 2
4.15	Added information on new iFrame version 4: `https://request.eprotect.vantivprelive.com/eProtect/js/eProtect-iframe-client4.min.js`. iFrame version 4 adds accessibility features including customizable error messages using the new `customErrorMessages` property.	Chapter 1, Chapter 2, and Appendix A
4.14	Removed information on jQuery as a required library for loading the eProtect client. eProtect uses plain JavaScript instead.	Chapter 1 and Chapter 2
4.13	Added example text for an Apple Pay POST Request.	Chapter 2
	Corrected spelling error in example text for `maskAfter`**`Success`**`Value`.	Chapter 2, Appendix A
4.12	Added a section, “Using eProtect ISO 8583, 610, and HHMI” with references to the appropriate guides to Chapter 1; removed the section, “Transaction Examples When Using ISO8583, 610, and HHMI” from Chapter 2.	Chapter 1 and Chapter 2
	Added information on `checkoutCombinedMode`, including a new section and code example.	Chapter 2
	Updated the flow description and figure in the section, “Using the Worldpay Mobile API for Google Pay” for clarity.	Chapter 2
4.11	Updated API sample code in Appendix A to include example for EBT/SNAP PAN length validation.	Appendix A
4.10	Added information on two new parameters (`minPanLength` and `maxPanLength`) allowing ability to adjust the EBT/SNAP PAN length validation.	Chapter 2, Appendix A
4.9	Re-worded various instructions in Apple Pay and Google Pay implementation for clarification.	Chapter 1
	Added information on an update to Font Awesome in iFrame (required due to updated Visa Logo).	Chapter 1
	Added new iFrame properties, `enhancedUxVersion` (related to new mandate for Visa logo) and `maskAfterSuccessValue` (related to previously-inputted values returned).	Chapter 2, Appendix A
4.8	Added two eProtect-specific response codes related to EBT/SNAP PIN numbers (Table 1-3).	Chapter 1
4.7	Added information on using the `pciNonsensitive` attribute to utilize MOD10 checking for private label cards (set to false).	Chapter 2

TABLE 1 Document Revision History (Continued)

Doc. Version	Description	Location(s)
4.6	For EBT/SNAP PIN, added a new property for PIN placeholder text, information Information on PIN error code handling, and information on EBT multi-tender payments.	Chapter 2
	Added new `encryptionKey` parameter for Visa Checkout along with a new pre-live testing site (https://www.testvantivcnp.com/checkout/checkout4VisaCheckout-prelive-sandbox.jsp).	Chapter 2
	Changed the maximum field length for `orderId` from 25 to 256, per an update of the cpnAPI (v12.19).	Chapter 2 and Appendix A
	Updated the ApplePay PKPayment token URL referencing Apple documentation.	Chapter 2
	Updated the cnpAPI version to 12.19.	All
4.5	Added section on timeout values; added references to the new section in other chapters and Appendix C.	Chapter 1, 2 and Appendix C
4.4	Removed PWS information due to end of product life.Added information on testing	All
	Added information in the testing section on the use of PCI non-sensitive values.	Chapter 2
	Updated flow graphics.	Chapter 1
4.3	Added information on new Checkout PIN and checkoutPinMode for iFrame and JavaScript API modes (for EBT/SNAP cards).	Chapter 2
	Added two new URLs for testing/certification of EBT PAN and PIN values.	Chapter 1 and 2
4.2	Corrected the 'cvv2' element in Table 2-5 and in Appendix A (as a `formField` variable).	Chapter 2, Appendix A
	Added the `location` element to the register token response element and example	Chapter 2, Appendix A
	Updated the cnpAPI version to 12.15.	All
4.1	Removed optional `pcinonsensitive` parameter from examples to prevent accidental use if cutting and pasting code from the guide.	Chapter 2, Appendix A
	Corrected part of the following line in sample code: `typeof eProtectiframeClient` was changed to `typeof EprotectIframeClient`	Chapter 2, Appendix A
	Updated for cnpAPI Rel. 12.12 (no changes affecting eProtect).	All

TABLE 1 Document Revision History (Continued)

Doc. Version	Description	Location(s)
4.0	Removed 'Enterprise' terminology from the guide due to the replacement of the Litle Vault with the Omnitoken solution (and the retirement of eCommerce eProtect Guide). Also folded in some sections and information from the eCommerce eProtect Guide.	All

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

The Worldpay eCommerce documentation set includes the items listed below:

- *Worldpay eComm iQ Reporting and Analytics User Guide*
- *Worldpay eComm cnpAPI Reference Guide*
- *Worldpay eComm Chargeback API Reference Guide*
- *Worldpay eComm Chargeback Process Guide*
- *Worldpay eComm PayFac API Reference Guide*
- *Worldpay eComm PayFac Portal User Guide*
- *Worldpay eComm cnpAPI Differences Guide*
- *Worldpay eComm Scheduled Secure Reports Reference Guide*
- *Worldpay eComm Chargeback XML and Support Documentation API Reference Guide* **(Legacy)**

Typographical Conventions

Table 2 describes the conventions used in this guide.

TABLE 2 Typographical Conventions

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
< >	Angle brackets are used in the following situations: • user-supplied values (variables) • XML elements
[ ]	Brackets enclose optional clauses from which you can choose one or more option.
bold text	Bold text indicates emphasis.
Italicized text	Italic type in text indicates a term defined in the text, the glossary, or in both locations.
blue text	Blue text indicates a hypertext link.

Contact Information

This section provides contact information for organizations within Worldpay.

Worldpay Contact Center - For technical issues related to eProtect in production for ISO 8583, 610 Interface, and other Core platforms issues.

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

Technical Support - For eCommerce technical issues such as file transmission errors, email Technical Support. A Technical Support Representative will contact you within 15 minutes to resolve the problem. For critical production issues, use the number listed below.

Technical Support Contact Information

Phone	*For critical production issues only*: 1-888-829-1907

Technical Support Contact Information

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

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

Relationship Management/Customer Service Contact Information

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

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

Chargebacks Department Contact Information

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

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

Technical Publications Contact Information

E-mail	TechPubs@fisglobal.com

worldpay

1

Introduction

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

1.1 eProtect Overview

Worldpay's eProtect and OmniToken solutions help solve your card-not-present challenges by virtually eliminating payment data from your systems. The eProtect solution reduces the threat of account data compromise by transferring the risk to Worldpay, reducing PCI applicable controls. No card data is actually transmitted via your web server.

When the card holder submits their account information, your checkout page calls the eProtect service to exchange the account number for a low-value token (the *Registration ID*). The Registration ID--a PCI non-sensitive value--is exchanged for a high value Worldpay token in a transaction (`authorization`, `sale`, or `registerToken`). The API call for the token exchange is completed within Worldpay processing environments and is not transmitted to the card networks. Therefore, CVV and AVS information is not checked, and the expiration date is not validated. Any credit card that passes MOD10 receives a Registration ID. The Worldpay token server stores only the card number, and does not include CVV, expiration date, or any other cardholder information.

Worldpay ensures high service availability for eProtect by internally implementing primary and secondary endpoint routing (i.e., internal routing to a secondary site if the primary site is unavailable). High availability is supported when using the eProtect JavaScript API V.3.or higher.

> **NOTE:** The Mobile API POST option does not use internal routing in Worldpay but instead utilizes a primary and secondary endpoint in our production environment (our pre-live environment only uses one endpoint). You can mirror our internal logic by connecting to the primary for five (5) seconds and if a response is not received from the primary endpoint in five (5) seconds, you can configure your system to connect to our secondary endpoint and allow 15 seconds before timing out.

Worldpay 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 Worldpay eProtect product, see the *Vantiv eProtect iFrame Technical Assessment Paper*, prepared by Coalfire IT Audit and Compliance.

Figure 1-1 illustrates eProtect with Omnitoken in the section, How eProtect Works next.

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

1.1.1 Using eProtect with ISO 8583, 610 and HHMI

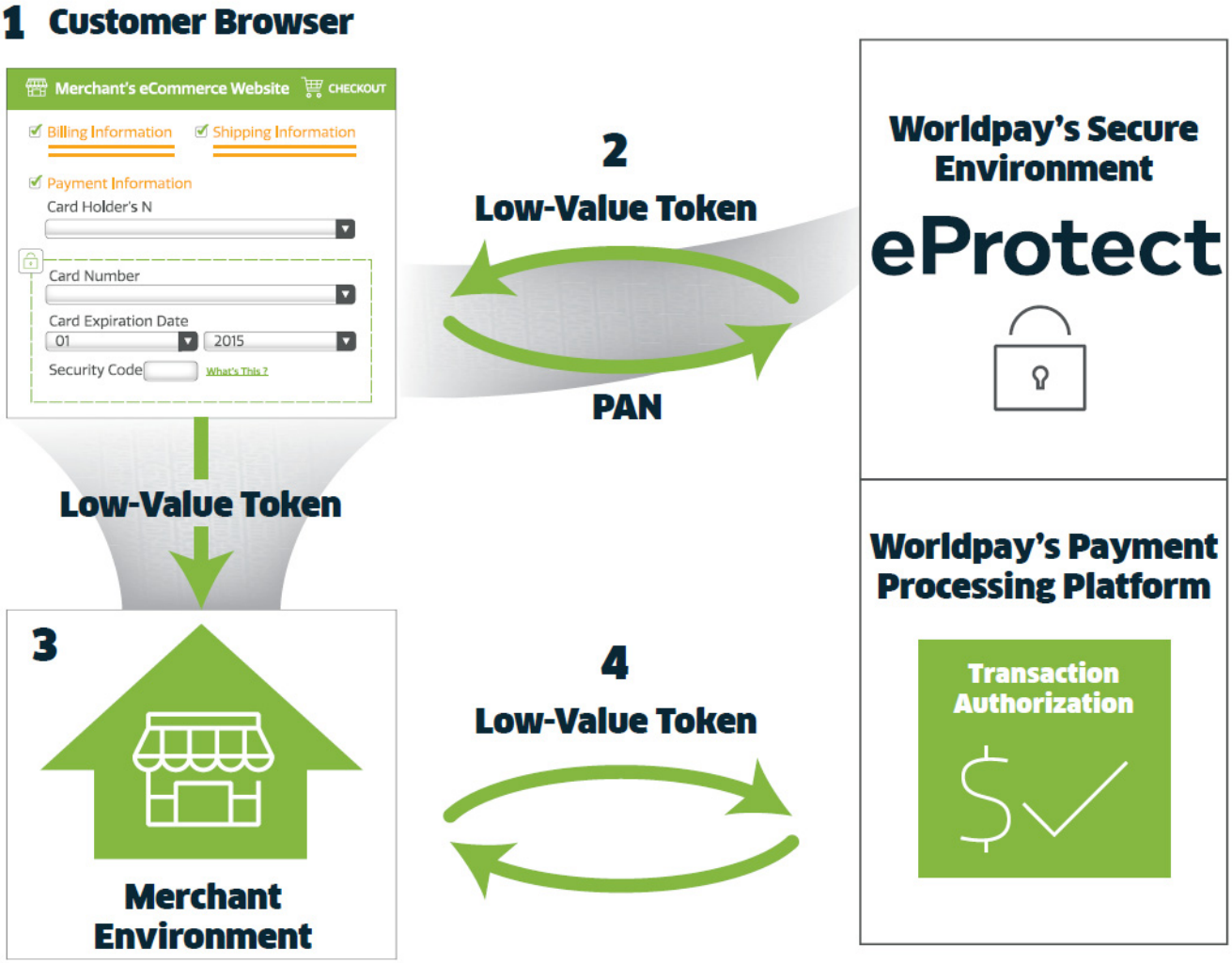
You can use eProtect for transactions using the ISO 8583, 610, and HHMI message interface specifications. These transactions are submitted by your payment processing system after your customer clicks the submit button on your checkout page. Your payment processing system sends the transactions to Worldpay with the `<paypageRegistrationId>` returned by eProtect and the Worldpay maps the Registration ID to the OmniToken and card number.

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

- *ISO 8583 Reference Guide*
- *610 Interface Reference Guide*

1.2 How ePprotect Works

FIGURE 1-1 ePprotect Process



1. The cardholder enters their details in the eProtect iFrame, hosted on Worldpay's dedicated eProtect server.
2. The eProtect server returns a single-use, low-value token to you. The payment information is forwarded to Worldpay's data security platform awaiting an authorization request.
3. You use the low-value token to process the order.
4. Once Worldpay receives authorization, we convert the low-value token to a high-value token – Worldpay's OmniToken – and return it to you. This high-value token contains the authorization response The OmniToken may be used for follow-on transactions, like card-on-file, returns, recurring billing, etc.

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 Worldpay for verification. See Creating a Customized CSS for iFrame on page 18 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 28.
  - Integrating iFrame into your Checkout Page on page 42.
  - Integrating eProtect Into Your Mobile Application on page 58.
- 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.

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 covers the use your own version of jQuery, as described within.

> **NOTE:** Starting from March 2022, jQuery is no longer a required library. eProtect uses plain JavaScript instead of jQuery.

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 Version 3 Hosted iFrame-Integrated Checkout Page on page 100.

1. Remove the script that was downloading `eProtect-api3.js`.
2. Add a script tag to download `eprotect-iframe-client4.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 Worldpay for verification. See Creating a Customized CSS for iFrame on page 18 and Configuring the iFrame on page 43 for more information.
8. See Calling the iFrame for the Checkout ID on page 48 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):

- Google Chrome 22 and later
- Mozilla Firefox 27 and later
- Microsoft - Internet Explorer 11 and later, Internet Explorer Mobile 11 and later, Edge 12 and later
- Safari 7 and later, Safari Mobile 6 and later
- Opera 14 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, Worldpay 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, Worldpay requires that you use TLS 1.2 encryption.

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

Worldpay 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 Worldpay Mobile API for Apple Pay on page 61 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 28 and Using the Customer Browser JavaScript API for Apple Pay on the Web on page 37 for more information on the complete process. Also, see Table 1-1, Apple Pay on the Web Compatible Devices for further information on supported Apple devices.

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

TABLE 1-1 Apple Pay on the Web Compatible Devices

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

1.3.4 ePrOtect Support for Google Pay™

Google Pay 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 Google Pay 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 Pay API to simplify payments for customers who make purchases in Android apps or on Chrome with a mobile device.

Worldpay supports two methods for merchants to submit Google Pay transactions from Mobile applications to the FIS-Worldpay platform. The preferred method involves you sending certain Worldpay-specific parameters to Google. The response from Google includes a Worldpay `paypageRegistrationId`, which you use normally in your payment transaction submission to Worldpay. With the alternate method, you receive encrypted information from Google, decrypt it on your servers, and submit the information to Worldpay in a payment transaction. See Using the Worldpay Mobile API for Google Pay on page 68 for more information.

1.3.5 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. Worldpay supports Visa Checkout purchases from your website or mobile app. initiated from compatible devices.

1.3.5.1 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) 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 Worldpay eProtect.

1.3.5.2 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 Worldpay**: During the on-boarding process, Worldpay Implementation assigns an `externalClientId` to denote the relationship between Worldpay, 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 39 and Using the Worldpay Mobile API for Visa Checkout on page 65.

1.3.6 jQuery Version

If you are implementing a browser-based solution, you have the option to use a jQuery library or JavaScript before loading the eProtect API. If using jQuery, we recommend using jQuery 1.4.2 or higher. Refer to http://jquery.com for more information on jQuery. If using JavaScript, simply leave out the reference to jQuery in your code.

1.3.7 Certification and Testing Environments

For certification and testing of Worldpay feature functionality, Worldpay uses the ***Pre-Live*** testing environment. The Pre-live environment is for all merchant certification testing for both new merchants on-boarding to Worldpay, 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)
- Version 3 iFrame Client (eprotect-iframe-client3.js)
- Version 4 iFrame Client (eprotect-iframe-client4.js
- iFrame JavaScript (eProtect-iframe.js)

> **NOTE:** iFrame Version 4 adds accessibility features including customizable error messages using the new `customErrorMessages` property. If you wish to migrate from iFrame Version 3 to iFrame Version 4, be aware that adjustments are required to your existing style sheet (see iFrame Accessibility on page 25), as well as error messages (see Handling Errors - iFrame Version 4 on page 55).

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 version 3	https://request.eprotect.vantivprelive.com/eProtect/js/eProtect-iframe-client3.min.js
	iFrame version 4	https://request.eprotect.vantivprelive.com/eProtect/js/eProtect-iframe-client4.min.js
	POST Request Submission (for Mobile API)	https://request.eprotect.vantivprelive.com/eProtect/paypage
	API Request (Visa Checkout only)	https://request.eprotect.vantivprelive.com/eProtect/s.json
	Tokenized EBT PAN Request (EBT/SNAP cards only)	https://www.testvantivcnp.com/checkout/checkout-ebt-noexp.html
	Tokenized EBT PIN Request (EBT/SNAP cards only)	https://www.testvantivcnp.com/checkout/checkout-pin.html
Live Production	*Production*	*Contact your Implementation Consultant for the eProtect Production URL.*

1.3.7.1 Pre-Live Environment Limitations and Maintenance Schedule

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

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

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

- Merchant profile and data is deleted after seven (7) consecutive days with no activity.
- Maintenance window (for Core platform) - every other Thursday from 10:00 PM to 6:00 AM ET
- Maintenance window (for all eCommerce pre-live environments) - each Tuesday and Thursday from 4:00 AM to 8:00 AM ET.
- Daily limit of 1,000 Online transactions.
- Daily limit of 10,000 Batch transactions.

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

1.3.8 Transitioning from Certification to Production

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

1.3.9 eProtect-Specific Response Codes

Table 1-3 lists response codes specific to the eProtect feature, received in the browser or mobile device, and those received via the applicable Worldpay 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.

> **NOTE:** Some user input error messages are customizable when using Version 4 of the iFrame solution. See Handling Errors - iFrame Version 4 on page 55 for more information.

TABLE 1-3 ePrototect-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	FIS-Worldpay
885	eProtect iFrame CSS failed load - <*number*>	System	FIS-Worldpay
889	Failure	System	FIS-Worldpay
893	PIN num too short	Validation	User
894	PIN num too long	Validation	User

NOTE: For information on response codes specific to OmniToken transactions, see the applicable Worldpay 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.

TABLE 1-4 ePrOtect Response Codes Received in cnpAPI Responses (Continued)

Response Code	Response Message	Response Type	Description
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.

1.3.10 eProtect Registration ID Duplicate Detection

Worldpay 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. Worldpay strongly recommends that the order ID and transaction ID data elements be unique for every registration request.

1.3.11 Setting Timeout Values

You configure timeout values in eProtect and iFrame configurations to assist in determining whether your checkout page was built successfully. If timeout values are set too low, the iFrame may fail to load. If timeout values are set too high, the secondary server may not have time to load.

Table 1-5 gives timeout value recommendation for the three request types. See Configuring the iFrame on page 43 for more information.

TABLE 1-5 Timeout Value Recommendations

Request Type and URL	99.9th Percentile Response Time *	Property and Definition	Recommendations
Client JavaScript that loads iFrame: `https://request.eprotect.vantivcnp.com/eProtect/js/payframe-client.min.js`	900ms (0.9 seconds)	--	Worldpay recommends that you **do not use any** internal timeout logic before loading the iFrame client JavaScript. Some slow devices (i.e., mobile phones, slow Internet connections) could take as long as 15 seconds to load the JavaScript. If your configuration does not allow the JavaScript to load, the iFrame will not render for customer payment input.
iFrame with PAN, expDate, or CVV: `https://request.eprotect.vantivcnp.com/eProtect/eProtect_`*`mypaypageIdnumber`*`.html`	900ms (0.9 seconds)	`htmlTimeout` (*Optional*) The amount of time (in milliseconds) to wait for the iFrame to load before responding with an '884' error code. If you receive an 884 code, the payment cannot proceed.	Use the default timeout value of **5000** (5 seconds). If you receive frequent '884' errors due to the iFrame failing to load, increase the `htmlTimeout` value.
Tokenization API call: `https://request.eprotect.vantivcnp.com/eProtect/paypage`	6000ms (6 seconds)	`timeout` (*Required*) The number of milliseconds before a transaction times out and the timeout callback in invoked. If the response from the primary server takes more than five (5) seconds, the request is automatically sent to our secondary server.	Set a timeout value of **15000** (15 seconds) to ensure the secondary server has time to respond.

* This value represents the observed timeout in 0.1% of requests (99.9th percentile). Table 1-6 lists the range of observed response times based on a sampling of JavaScript and HTML file retrievals

(approximately 10,000,000 requests) and API calls (approximately 40,000,000 requests) over a 30-day period.

TABLE 1-6 Observed Response Times - 30 Day Sample

Description	**Median**	**10% of Requests** (90th percentile)	**1% of Requests** (99th percentile)	**0.1% of Requests** (99.9th percentile)
Client JavaScript that loads iFrame	80ms	300ms	700ms	900ms
iFrame with PAN/expDate/CVV	5ms	80ms	400ms	900ms
Tokenization API call	400ms	1060ms	2000ms	6000ms

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 Worldpay, 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 Worldpay

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

For a list of allowable and disallowed CSS Properties, see .Appendix B, "CSS Properties for iFrame API".

1.4.1 CSS iFrame Validation and Customization Features

Worldpay 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 43 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).

  For mobile users, this is automatically facilitated by the presentation of a telephone pad when entering these fields, rather than the standard alphanumeric board.
- *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 Support* - Worldpay supports a hosted font library for standard web fonts. You specify the 'font-family' property in your CSS. Worldpay does not support custom fonts.
- *Icons* - Worldpay hosts an industry-standard icon library, SVG Icons (Font Awesome, v4.7.0) on our servers for you to leverage in your CSS. eProtect uses Font Awesome in iFrame primarily for payment card brand icons.

> **NOTE:** You must use Font Awesome v5.5.13 when using the Visa logo on your iFrame checkout page. To upgrade to V5.5.13, pass the configuration property `enhancedUXVersion` (an option of `enhancedUxFeatures`) with a value of `2` to obtain version 5.15.3. Any other value passed or lack of parameter results in continued use of version 4.7.0. See Configuring the iFrame on page 43 for more information.

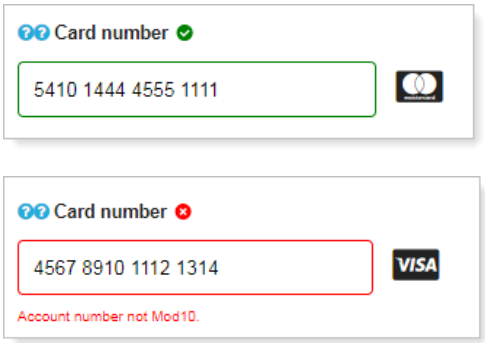
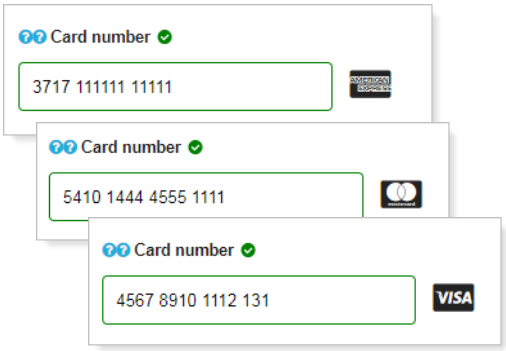
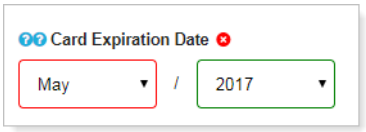
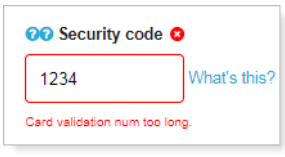
- *Trust Badge* - you can add a 'trust' badge (e.g., a padlock or shield icon) on the payment form to reassure your customers that your site is legitimate and that all their personal data is collected securely through trusted third-party service providers. Note that the trust badge can be displayed *in place of* the card graphic; your page cannot display both.

> **NOTE:** Worldpay does not offer the option to mask inputted values, as the inputs are hosted in the PCI-compliant iFrame environment. Masking the inputs does not add any additional security value.

Table 1-7, "iFrame Checkout Page Customizations - In-Line Field Validations" and Table 1-8, "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 1-7 are all included.

TABLE 1-7 iFrame Checkout Page Customizations - In-Line Field Validations

Field	Validation Behavior	Samples
Card Number	The iFrame checks the card number for correct size (too short or too long) and against the Luhn/Mod10 algorithm. In this example, if the consumer's inputs are valid, you can configure the iFrame to display green field borders and include a green check mark. Red borders and a red 'X' can indicate invalid input. The frame colors are customizable in your style sheet. The error messages are also customizable either via your style sheet or through JavaScript, depending on the version of the eProtect iFrame solution you are using. See Handling Errors - iFrame Version 4 on page 55 for more information.	
	The iFrame identifies the card type (Visa, Mastercard, Amex, etc.) based on the first few digits entered, and displays the appropriate card graphic. If the card type is unknown, the iFrame displays a generic card graphic. You can configure your style sheet to hide the card graphic. In addition, the iFrame auto-formats the arrangement of the card digits based on the initial entry.	
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 1-8 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 1-8 Style Sheet and iFrame Customizations

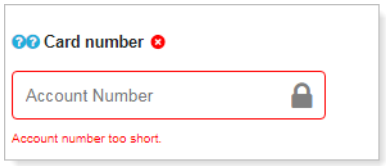
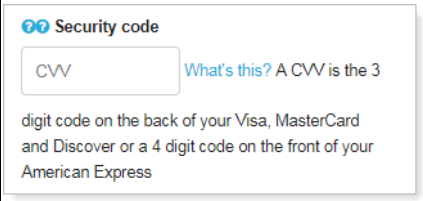
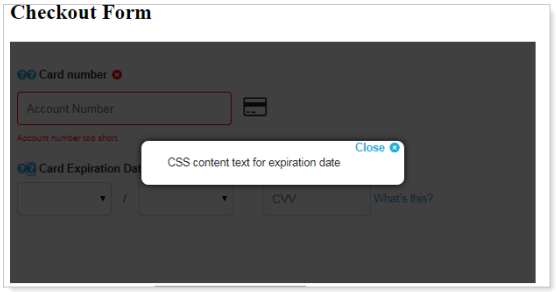
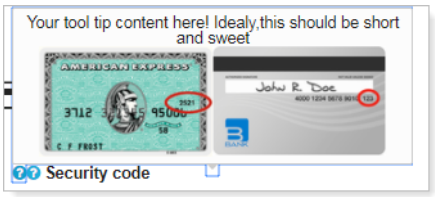
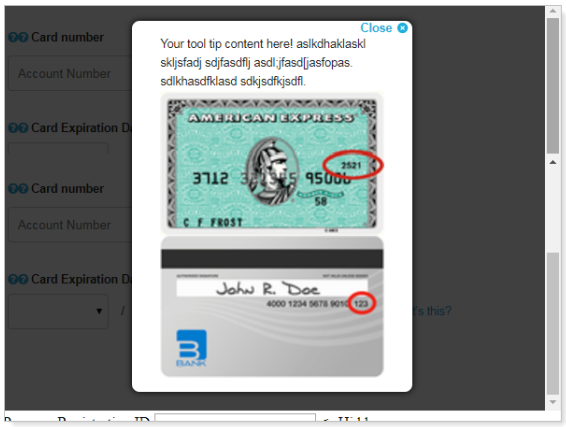
Customization	Samples
Trust Badge - You can add a 'trust badge' (e.g., a padlock or shield icon) to the payment form, using the Font Awesome (V4.7.0) icon library. Note that the trust badge can be displayed *in place of* the card graphic; your page cannot display both.	
Tool Tips - you control the following tool tip behavior in your style sheet:	
You can add a tool tip for any field (not just security code) activated by hovering, or when clicking 'What's This?'	 *Tool tip displayed after clicking 'What's This?'*
You can configure your style sheet to activate a tool tip by hovering over the '?' icon (rather than clicking). This is useful for short statements. You can also configure a modal dialog to activate on the click of the second '?' icon to display more lengthy CSS content.	 *Modal dialog displayed upon clicking second '?' icon.*

TABLE 1-8 Style Sheet and iFrame Customizations (Continued)

Customization	Samples
Tool Tips (*continued*) You can configure your CSS to display a Security code modal dialog where the tool tip displays generic card art showing the placement of CVC on cards. You can hide this with the CSS, if you choose. You can also remove the scrollbars, as well as direct your CSS to auto-size the dialog based on content.	  *Modal dialog displayed upon clicking first or second '?' icon at the security code field.*

1.4.2 Using Web Developer Tools

By using standard browser-provided web developer tools, you can develop and customize your CSS prior to sending it to Worldpay 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="tel" 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 Worldpay Implementation Consultant for review.

1.4.3 Reviewing your CSS with Worldpay

Worldpay 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, Worldpay 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 Worldpay 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)`, Worldpay 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 18, use the following URL:

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

1.4.3.1 CSS Properties Not Listed

There may be properties not listed in Appendix B, "CSS Properties for iFrame API" that you wish to use when creating your style sheet. We do not list every non-allowed CSS property, just those that we explicitly black-list (or that are 'excluded from the white-list'). There may be an opportunity to evaluate new CSS properties to add to the white-list. Please contact your Implementation Consultant to initiate a request for future development consideration of CSS properties.

1.5 iFrame Accessibility

eProtect iFrame Version 4 includes features that improve accessibility for HTML and error messages including:

- Expiration date fields grouped and labeled in a way that is more understandable to screen-reading programs.
- A mechanism for alerting customers to errors caused by invalid input (such as a card number that is too short).

eProtect iFrame Version 3 is still available but does not contain these accessibility upgrades. In Version 3, a CSS class is added to the input field containing the error, and this class can be used to style the field or to display an error message into the iFrame using the CSS `::before` and `::after` pseudo-elements. Screen readers cannot convey styling information and cannot detect content inserted using these CSS pseudo-elements.

eProtect Frame Version 4 inserts customizable error messages into the document structure using JavaScript. These error messages are detected and read by screen readers.

TABLE 1-9 eProtect iFrame Versions

iFrame Version	Client File Name	Accessible HTML	Accessible Error Message
4	eProtect-iframe-client4.min.js	Yes	Yes
3	eProtect-iframe-client3.min.js	No	No

For more information, see Handling Errors - iFrame Version 3 on page 55 and Handling Errors - iFrame Version 4 on page 55.

worldpay

2

Integration and Testing

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

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

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 58 for more information on the mobile solution.

See Integrating iFrame into your Checkout Page on page 42 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:

- **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 96.

2.1.2 Loading the eProtect API and jQuery

> **NOTE:** Starting from March 2022, jQuery is no longer a required library. eProtect uses plain JavaScript instead of jQuery.

You have the option to have your checkout page load a version of the jQuery JavaScript library before loading the eProtect client JavaScript library. To load the eProtect client JavaScript library from the eProtect application server to your customer's browser, insert the JavaScript below into your checkout page.

> **NOTE:** To avoid disruption to transaction processing, Worldpay 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). Worldpay 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 `reportGroup` field is required however it is not used for eProtect integrations on the Core platform. Use any value from 1-25 characters.

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
`ccNum`	(*Optional*) The credit card account number. Not applicable when `checkoutIdMode` is set to true.
`cvv2Num`	(*Optional*) The card validation number, either the CVV2 (Visa), CVC2 (Mastercard), or CID (American Express and Discover) value.
`paypageRegistrationId`	(*Required*) The temporary identifier used to facilitate the mapping of a token to a card number. Not applicable when `checkoutIdMode` is set to true.
`checkoutId`	*(Optional)* The low-value token ID exchanged for the CVV value, when `checkoutIdMode` 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.)

TABLE 2-1 `eProtectFormFields` Definitions (Continued)

Field	Description
`bin`	(*Optional*) The bank identification number (BIN), which is the first six digits of the credit card number. Not applicable when `checkoutIdMode` is set to true.
`pin`	(*Optional*) The PIN to be tokenized, when `checkoutPinMode` is set to true. For use with EBT/SNAP cards only.
`pinCheckoutId`	(*Optional*) The low-value token ID exchanged for the PIN value, when `checkoutPinMode` is set to true. For use with EBT/SNAP cards only.

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`, `bin`, `code`, `message`, `responseTime`, `type`, `vantivTxnId`, `firstSix`, `lastFour`, and `accountRangeId`. You have flexibility in the naming of these fields.

NOTE: The `accountRangeId` is only seen by merchants enabled for Issuer Insights, a Worldpay eCommerce Value-added Service.

```
<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"/>
  <input type="hidden" id="response$accountRangeId" name="response$accountRangeId"
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 have the option to add a jQuery selector to handle the submission `click` JavaScript event. The addition of the `click` event creates a eProtect Request and calls `sendToEprotect`.

The `sendToEprotect` call includes a timeout value in milliseconds. 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). See Setting Timeout Values on page 15 for additional information.

> **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 applepay = {};
        applepay.data = "",
        applepay.signature = "";
        applepay.version = "";
        applepay.header = {};
        applepay.header.applicationData = "";
        applepay.header.ephemeralPublicKey = "";
        applepay.header.publicKeyHash = "";
        applepay.header.transactionId = "";
      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"
       "minPanLength" : 16

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

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

TABLE 2-2 `eProtectRequest` Fields

Field	Description
`paypageId`	(*Required*) The unique number assigned by Implementation.
`reportGroup`	(*Required*, *but not used by eProtect integrations on the Core platform.)* The cnpAPI attribute that defines under which merchant sub-group this transaction will be displayed in eCommerce iQ Reporting and Analytics.
`orderId`	The merchant-assigned unique value representing the order in your system (used when linking authorizations, captures, and refunds, and for retries). Worldpay recommends that the values for `id` and `orderId` be different and unique so that we can use these identifiers for reconciliation or troubleshooting. If you do not have the order number available at this time, please generate another unique number to send as the `orderId` (and send it to your servers to map it to the order number that you generate later).
`id`	The merchant-assigned unique value representing this transaction in your system. The same value must be used for retries of the same failed eProtect transaction but must be unique between the eProtect transaction, authorization, capture, and refund for the same order. Worldpay recommends that the values for `id` and `orderId` must be different and unique so that we can use these identifiers for reconciliation or troubleshooting.
`checkoutIdMode`	*(Optional)* Determines whether `checkoutId` mode is activated. Setting the value to true causes only the cvv2 form field to be exchanged with eProtect, and returns a `checkoutId` upon a successful callback (instead of the `paypageRegistrationId`).
`checkoutPinMode`	*(Optional)* Determines whether `checkoutPinMode` is activated. Setting the value to true causes only the PIN form field to be exchanged with eProtect, and returns a `pinCheckoutId` upon a successful callback (instead of the `paypageRegistrationId`). Do not use with `checkoutIdMode`. For use with EBT/SNAP cards only.
`applepay`	(*Optional*) The Apple Pay PKPaymentToken. Required for Apple Pay on the Web. Table 2-8 on page 63 describes the Apple Pay components.
`url`	(*Required*) The URL to request submission for eProtect. See Table 1-2, eProtect Certification, Testing, and Production URLs on page 12.
`minPanLength`	(*Optional*) Minimum number of digits that must be present in the customer-supplied PAN value. Defaults to 13 if this value is not provided.
`maxPanLength`	(*Optional*) Maximum number of digits allowed in the customer-supplied PAN value. Defaults to 19 if this value is not provided.

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$accountRangeId').value =  response.accountRangeId;
  document.getElementById('response$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 31. We recommend a timeout value of 15000 (15 seconds). See Setting Timeout Values on page 15 for more information.

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)" );
</script>
...
</HTML>
```

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

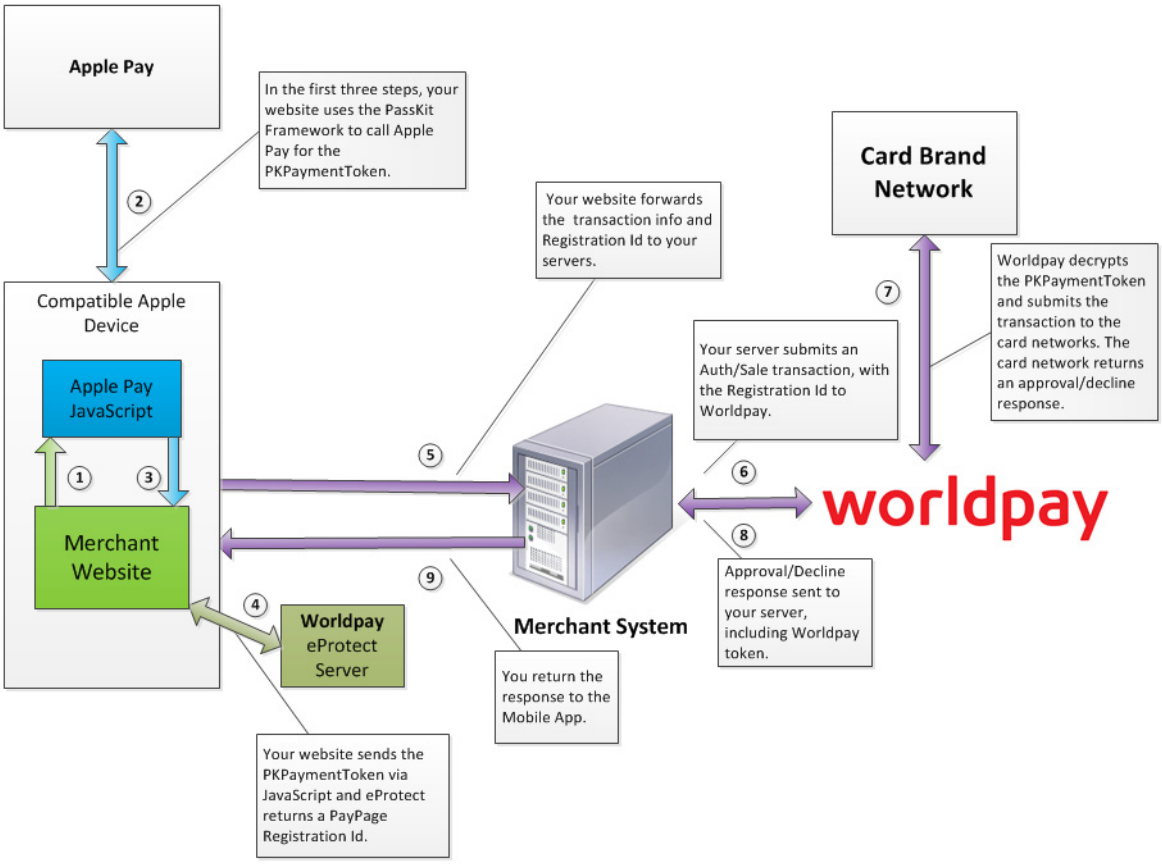
2.1.9 Using the Customer Browser JavaScript API for Apple Pay on the Web

In this scenario, the Worldpay 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 *Worldpay 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/documentation/passkit/pkpaymenttoken) to your application.
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, Worldpay decrypts the PKPaymentToken associated with the Registration ID and submits the transaction with the appropriate information to the card networks for approval.
8. Worldpay sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response and includes the Worldpay 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 Worldpay API key or Encryption 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) to your website.

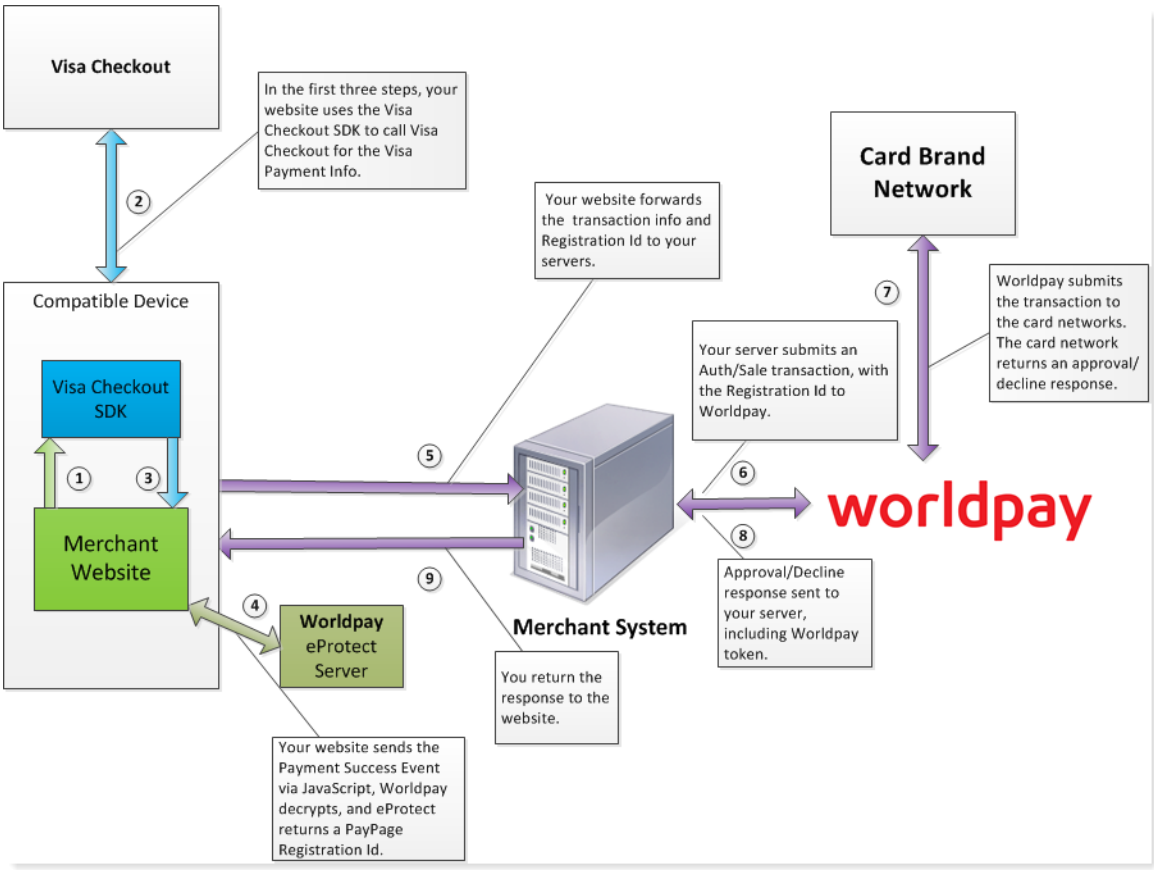
In this scenario, the Worldpay eProtect Customer Browser JavaScript API controls the fields on your checkout page that hold sensitive card data. When the cardholder clicks the Visa Checkout button, communication is exchanged with Visa Checkout via the JavaScript API to obtain the Payment Success event.

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 Worldpay 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/submits an Authorization/Sale transaction using the Registration ID.
7. Worldpay submits the transaction with the appropriate information to the card networks for approval.
8. Worldpay sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response and includes the Worldpay 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 - Worldpay 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, Encryption Key, and External Client ID
- Sending Worldpay the Required Fields

2.1.11.1 Requesting and Configuring the API Key, Encryption Key, and External Client ID

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();
    V.init({
            apikey: ep.getVisaCheckoutApiKey(), //Worldpay's Visa Checkout API Key
            encryptionKey:ep.getVisaCheckoutEncryptionKey(), //Worldpay's Encryption key
            sourceId: "Merchant Defined Source ID",
            externalClientId: "stefan_sandwiches", //Merchant client id - get this from
    Worldpay implementations team
            settings: {
                        ...
            },
            paymentRequest: {
                    ...
    }
    });
     ...
    }
    </script>
```

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

2.1.11.2 Sending Worldpay the Required Fields

Insert the following to send the required fields to Worldpay:

```
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 (including `encryptionKey`), go here:

https://www.testvantivcnp.com/checkout/checkout4VisaCheckout-prelive-sandbox.jsp.

For an example of a completed checkout page without `encryptionKey`, go here:

https://www.testvantivcnp.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 18 for information on creating a style sheet. Also see https://www.testvantivcnp.com/iframe/ to view our iFrame example page.

2.2.1 Integration Steps

Integrating the iFrame into you checkout page includes the following steps, described in the sections to follow. For a full HTML code examples of iFrame eProtect implementations, see the HTML Checkout Page Examples on page 96.

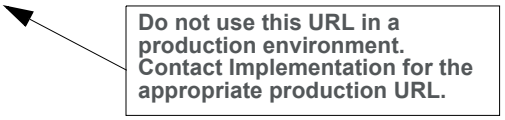
1. Loading the iFrame
2. Configuring the iFrame
3. Calling the iFrame for the Registration ID
4. Handling Callbacks

> **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-
client4.min.js"></script>
```



2.2.3 Configuring the iFrame

To configure the iFrame after the page is loaded, you specify the required properties listed in Table 2-3 (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`. The variable `iframeIsReady` in the `checkPayframeLoaded` function determines whether the iFrame is rendered so you can unhide the `div`.

```
function ready(callback) {
  // in case the document is already rendered
  if (document.readyState != 'loading') callback();
  // modern browsers
  else if (document.addEventListener) document.addEventListener('DOMContentLoaded', callback);
  // IE <= 8 for browser's not supporting addEventListener property
  else document.attachEvent('onreadystatechange', function() {
    if (document.readyState == 'complete') callback();
  });
}
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,
    "maskAfterSuccessValue": 'Z',
    "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",
      "pin":"PIN Placeholder"
    },
```

```
      "inputsEmptyCallback": inputsEmptyCallback,
      "enhancedUxFeatures" : {
        "inlineFieldValidations": true,
        "expDateValidation": false,
        "enhancedUxVersion": 2
        }
    "minPanLength": 16,
    "iFrameTitle":"My Custom Title",
    "label" : {
      "accountNumber":"Account Number",
      "expDate" : "Exp Date",
      "cvv" : "CVV",
    "pin":"Pin"
    },
  };
  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);

  function checkPayframeLoaded(){
      if(iframeIsReady===true){
          //code changes
      }
    };
    checkPayframeLoaded();

  eProtectiframeClient.autoAdjustHeight();
});

window.onmessage = function(event) {
  if(event.data === "checkoutWithEnter") {
      //Captures Enter event from iFrame
    var message = {
             "id": document.getElementById("request$merchantTxnId").value,
             "orderId": document.getElementById("request$orderId").value
          };
          startTime = new Date().getTime();
          payframeClient.getCheckoutPin(message);
          return false;
        }
      };
```

TABLE 2-3 Common Properties

Property	Description
`paypageId`	(*Required*) The unique number assigned by Implementation.
`style`	(*Required*) The CSS filename (excluding the '.css'). For example, if the style sheet filename is `mysheet1.css`, the value for this property is `mysheet1`.

TABLE 2-3 Common Properties (Continued)

Property	Description
`reportGroup`	(*Required*, *but not used by eProtect integrations on the Core platform.)* The cnpAPI attribute that defines under which merchant sub-group this transaction will be displayed in eCommerce iQ Reporting and Analytics.
`timeout`	(*Required*) The number of milliseconds before a transaction times out and the timeout callback in invoked. 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, Worldpay recommends a timeout value of 15000 (15 seconds).
`div`	(*Required*) The ID of the HTML `div` element where our iFrame is embedded as innerHTML.

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

`checkoutWithEnter`	*(Optional)* Determines whether the checkout function is activated (the outer frame is able to receive the Enter key message event) when pressing the Enter key within the iFrame. This enables you to trigger the form submit. This functionality is available for Account number, CVV, and PIN. See Capturing the Enter Event from the iFrame for more information.
`months`	(*Required*) Determines how the `expMonth` property is displayed (customizable).
`checkoutIdMode`	*(Optional)* Determines whether `checkoutIdMode` is activated. Set the value to ***true*** to establish a rule allowing the capture of the CVV only (in order to receive the `checkoutId`). Adding this field hides all fields in the iFrame, except CVV and CVV-related fields (including tooltips, etc.).
`checkoutPinMode`	(*Optional*) Determines whether `checkoutPinMode` is activated (for use with EBT/SNAP cards only). Set the value to ***true*** to establish a rule allowing the capture of the PIN only (in order to receive the `pinCheckoutId`). Adding this field hides all fields in the iFrame, except PIN. Do not use with `checkoutIdMode` or `checkoutCombinedMode`.

TABLE 2-3 Common Properties (Continued)

Property	Description
`checkoutCombinedMode`	(*Optional*) Determines whether `checkoutCombinedMode` is activated (for use with EBT/SNAP cards only). Set the value to ***true*** to establish a rule allowing the capture of the account number and PIN at the same time (to be exchanged for `paypageRegistrationId` and `pinCheckoutId`, respectively). Adding this field hides all fields in the iFrame, except account number and PIN. Do not use with `checkoutIdMode` or `checkoutPinMode`.
`inputsEmptyCallback`	*(Optional)* When a consumer returns to your checkout page to edit non-payment information, this function determines whether the Card number and security code fields are empty, and indicates whether to return this information in your callback. See Creating a Customized CSS for iFrame *on page 18* for more information.
`inlineFieldValidations`	*(Optional)* An option of `enhancedUxFeatures`.Determines whether in-field validations are performed (set value to `true`). See Creating a Customized CSS for iFrame *on page 18* for more information.
`enhancedUxVersion`	(*Optional*) An option of `enhancedUxFeatures`. Links to v5.5.13 of the Font Awesome stylesheet (mandatory when using the Visa logo on your checkout page). Set the value to `2` to obtain version 5.15.3. Any other value passed or lack of parameter results in continued use of Font Awesome version 4.7.0.
`height`	(*Optional*) The height (in pixels) of the iFrame. There are three options: • You can pass `height` as an optional parameter when configuring the client. • You can call `autoAdjustHeight` in the client to tell the iFrame to adjust the height to exactly the number of pixels needed to display everything in the iFrame without displaying a vertical scroll bar (recommended). **Note**: some browsers may not support this option. • You can ignore `height`. The iFrame may display a vertical scroll bar, depending upon your styling of the `div` containing the iFrame.
`noScrollBar`	(*Optional*) Determines whether to disable the vertical scrollbar in the iFrame (set value to `true`). If this property is omitted (existing default behavior), the iFrame shows the scrollbar as needed. Set this property value to `true` to make the scrollbar inside the iFrame permanently disabled.
`htmlTimeout`	(*Optional*) The amount of time (in milliseconds) to wait for the iFrame to load before responding with an '884' error code. The default timeout value is **5000** (5 seconds). If you receive frequent '884' errors due to the iFrame failing to load, increase the `htmlTimeout` value.

TABLE 2-3 Common Properties (Continued)

Property	Description
`maskAfterSuccessValue`	(*Optional*) Sets values previously inputted and returned in the iFrame to default values. Set to a single character to mask the PAN and CVV with the character. When the value is not set (default) or set with more than one character, the PAN is masked with 'X' except the last 4 digits; the CVV is masked with 'XXX.' When the value is blank, the PAN and CVV values are cleared.
`minPanLength`	(*Optional*) Minimum number of digits that must be present in the customer-supplied PAN value. Defaults to 13 if this value is not provided.
`maxPanLength`	(*Optional*) Maximum number of digits allowed in the customer-supplied PAN value. Defaults to 19 if this value is not provided.
`customErrorMessages`	(*Optional - iFrame Version 4 only*) Determines the custom error messages to display for input errors. The object keys are the error codes listed in Table 2-5, "Default Error Messages". If an error code is omitted, the default value displays. Use `null` to display no error message for a specific error code. See Handling Errors - iFrame Version 4 on page 55 for more information.
`iFrameTitle`	(*Optional*) Specifies a custom title for the iFrame. If you omit this property (default behavior), the iFrame shows the default value *Secure Card Data Capture*.
`label`	(*Optional*) Specifies a custom label for card number, expiration date, CVV, and PIN. If you omit this property (default behavior), the iFrame shows the following default values: **Card Number**: *Card number* **Exp. date**: *Card Expiration Date* **CVV**: *Security code* **PIN**: *Pin*

2.2.4 Capturing the Enter Event from the iFrame

You can configure your checkout page to receive the enter key message event when the shopper presses the enter key within the iFrame. This enables the outer frame to receive the enter key message event and trigger the form submit event handler. This functionality is available for Account number, CVV, and PIN.

There are two configuration options:

Option 1:

```
window.onmessage = function(event) {
  if(event.data === "checkoutWithEnter") {
    var message = {
             "id": document.getElementById("request$merchantTxnId").value,
             "orderId": document.getElementById("request$orderId").value
           };
           startTime = new Date().getTime();
           payframeClient.getCheckoutPin(message);
           return false;
       }
    };
```

Option 2:

```
window.addEventListener("message", function(event) {
  if(event.data === "checkoutWithEnter") {
    var messageToSend = {
             "id": document.getElementById("request$merchantTxnId").value,
             "orderId": document.getElementById("request$orderId").value
           };
           startTime = new Date().getTime();
           payframeClient.getCheckoutPin(messageToSend);
           return false;
    }
  });
```

2.2.5 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 an 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,
    };
  eProtectiframeClient.getPaypageRegistrationId(message);
  return false;
};
```

2.2.6 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`	The merchant-assigned unique value representing this transaction in your system. The same value must be used for retries of the same failed eProtect transaction but must be unique between the eProtect transaction, authorization, capture, and refund for the same order. **Type**: String **Max Length**: 25 characters Worldpay recommends that the values for `id` and `orderId` must be different and unique so that we can use these identifiers for reconciliation or troubleshooting.
`orderId`	The merchant-assigned unique value representing the order in your system (used when linking authorizations, captures, and refunds, and for retries). **Type**: String **Max Length**: 25 characters Worldpay recommends that the values for `id` and `orderId` be different and unique so that we can use these identifiers for reconciliation or troubleshooting. If you do not have the order number available at this time, please generate another unique number to send as the `orderId` (and send it to your servers to map it to the order number that you generate later).
`pciNonSensitive`	(*Optional*) Bypasses 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. A value of `false` allows certain methods of payment other private label cards to make use of the MOD10 check and return the BIN (first 6 digits of card number). When you set the `pciNonSensitive` parameter to `false`, the type attribute is not returned (because the method of payment cannot be determined) and does not cause a validation failure. ***Note***: If you use this parameter with a value of `true`, the card type and BIN are not returned in the response. See Notes on the PCI Non-Sensitive Value Feature, next.

2.2.6.1 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, JCB, and EBT/SNAP cards. The eProtect PCI Non-Sensitive

feature has the capability to capture and tokenize non-network branded payment types, such as private label and Worldpay gift cards. The PCI Non-Sensitive feature can tokenize 13-19 digit values, as long as the Worldpay message interface specification supports that payment type.

> **NOTE:** The PCI Non-Sensitive Value feature is designed for account numbers, and is therefore not applicable for sensitive authentication data requests (such as PIN, CVV2, etc.).

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.

2.2.7 Calling the iFrame for the Checkout PIN

Your checkout page can call the iFrame to exchange the PIN value for a `pinCheckoutId`, a low-value-token with a 24-hour lifespan. Use this when you store the high-value token on file for a consumer, but still want that consumer to populate the PIN with each eProtect transaction. See the parameters listed in Table 2-4 for more information.

In the case of an EBT multi-tender payment, if other forms of payment(s) are declined, reload the EBT PIN iFrames to clear out past values and prompt the cardholder to re-enter their EBT card information. If the EBT PIN low value token (LVT) was obtained in the original flow but not used, the point of sale software can choose to use the LVT PIN and simply hide the PIN iFrame. This reduces the number of low value tokens obtained as well as cost of using the service.

> **NOTE:** The `pinCheckoutId` is for use with EBT/SNAP cards only.

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

```
var message = {
    "id": document.getElementById("request$merchantTxnId").value,
    "orderId": document.getElementById("request$orderId").value
};
startTime = new Date().getTime();
iframeClient.getCheckoutPin(message);
```

To view a sample page and to test submission of a PIN value, go here:

https://www.testvantivcnp.com/checkout/checkout-pin.html

2.2.8 Calling the iFrame for the Registration ID and Checkout PIN

For added convenience, your checkout page can call the iFrame to exchange the account number (PAN) for a `paypageRegistrationId` *and* the PIN value for a `pinCheckoutId` at the same time. To do this, use the `checkoutCombinedMode` option when you configure the iFrame. (See the parameters listed in Table 2-3 for more information).

For security reasons, the iFrame makes two separate requests to eProtect: one to obtain the Registration ID and one to obtain the Checkout PIN. Each of these requests receives its own `vantivTxnId`. Use the `getCombinedTokens` function to call the iFrame to obtain the Registration ID and Checkout PIN.

> **NOTE:** The `checkoutCombinedMode` is for use with EBT/SNAP cards only.

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

```
var message = {
    "id": document.getElementById("request$merchantTxnId").value,
    "orderId": document.getElementById("request$orderId").value
};
startTime = new Date().getTime();
iframeClient.getCombinedTokens(message);
```

To view a sample page and to test submission of a PIN value and PAN value at the same time, go here:

https://www.testvantivcnp.com/checkout/combined-tokens.html

2.2.9 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). The `accountRangeId` in the callback is only seen by merchants enabled for Issuer Insights, a FIS-Worldpay Value-added Service.

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$type').value = response.type;
    document.getElementById('response$accountRangeId').value = response.accountRangeId;
    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, and paypageId 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 {
      //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
        }
      }
    };
```

2.2.9.1 Handling Callbacks When Using checkoutCombinedMode

When using `checkoutCombinedMode`, the callback response container holds two separate response objects:

- a response for the account number (PAN) tokenization request
- a response for the PIN tokenization request.

Assuming you name the parameter received by callback function `responses`:

- the response for the PAN tokenization request is held in `responses.panResponse` and contains the eProtect Registration ID (`paypageRegistrationId`).
- the response for the PIN tokenization request is held in `responses.pinResponse` and contains the low-value PIN token (`pinCheckoutId`).
- `responses.panResponse` and `responses.pinResponse` each contain all the fields that a regular response contains, including their own separate `vantivTxnId` values.

Handling checkoutCombinedMode Errors

If an error occurs before the iFrame finishes loading, the response includes only one object. This may happen if the iFrame HTML fails to load. In this case, `responses.panResponse` and `responses.pinResponse` are undefined, and `responses.response` contains the error code.

The two requests are independent of one another, and it is possible that:

- both requests succeed,
- both requests timeout or fail, or
- one request succeeds and the other request fails.

If only one request fails, we suggest that you add logic to re-submit only the PAN or PIN for tokenization. This reduces the number of low-value tokens obtained as well as the cost of using the service.

To resubmit the PIN only:

- Reload the iFrame and configure it with `checkoutPinMode` set to ***true***.

To resubmit the PAN only:

- Ensure that the `showCvv` is ***not*** set to **true** when configuring the iFrame.
- Use the CSS or jQuery to hide the Expiration Date drop-down.

```
var eProtectiframeClientCallback = function(responses) {
  if (responses.response && responses.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, and paypageId 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 (!responses.panResponse || !responses.pinResponse ||
         (responses.panResponse.timeout && responses.pinResponse.timeout)){
    // Malformed response or both PIN and PAN requests timed out (eProtect system is completely unavailable)
      alert("We are experiencing technical difficulties. Please try again or call us to complete your
order");
    //You may also want to log the customer ip, user agent, and time for debugging
}
else {
  // There are two responses to process (PIN and PAN)

  if (responses.panResponse.response){
    // Extract the PAN tokenization response
    document.getElementById('panResponse$code').value = responses.panResponse.response;
    document.getElementById('panResponse$message').value = responses.panResponse.message;
    document.getElementById('panResponse$responseTime').value = responses.panResponse.responseTime;
    document.getElementById('panResponse$targetServer').value = responses.panResponse.targetServer;
    document.getElementById('response$paypageRegistrationId').value =
responses.panResponse.paypageRegistrationId;
    document.getElementById('panResponse$vantivTxnId').value = responses.panResponse.vantivTxnId;
    document.getElementById('panResponse$merchantTxnId').value = responses.panResponse.id;
    document.getElementById('panResponse$orderId').value = responses.panResponse.orderId;
    document.getElementById('panResponse$reportGroup').value = responses.panResponse.reportGroup;
    document.getElementById('panResponse$type').value = responses.panResponse.type;
    document.getElementById('panResponse$lastFour').value = responses.panResponse.lastFour;
    document.getElementById('panResponse$firstSix').value = responses.panResponse.firstSix;
    document.getElementById('bin').value = responses.panResponse.bin;
    document.getElementById('paypageRegistrationId').value = responses.panResponse.paypageRegistrationId;
}

  if (responses.pinResponse.response) {
```

```
    // Extract the PIN tokenization response
    document.getElementById('pinResponse$code').value = responses.pinResponse.response;
    document.getElementById('pinResponse$message').value = responses.pinResponse.message;
    document.getElementById('pinResponse$responseTime').value = responses.pinResponse.responseTime;
    document.getElementById('pinResponse$targetServer').value = responses.pinResponse.targetServer;
    document.getElementById('response$pinCheckoutId').value = responses.pinResponse.pinCheckoutId;
    document.getElementById('pinResponse$vantivTxnId').value = responses.pinResponse.vantivTxnId;
    document.getElementById('pinResponse$merchantTxnId').value = responses.pinResponse.id;
    document.getElementById('pinResponse$orderId').value = responses.pinResponse.orderId;
    document.getElementById('pinResponse$reportGroup').value = responses.pinResponse.reportGroup;
    document.getElementById('pinCheckoutId').value = responses.pinResponse.pinCheckoutId;
}

  if (responses.panResponse.response &&  responses.panResponse.response == '870' &&
      responses.pinResponse.response &&  responses.pinResponse.response == '870') {
    // Both PAN and PIN tokenizations succeeded. Submit the form
}
  else if (responses.panResponse.response && responses.panResponse.response == '870') {
    // The PAN tokenization succeeded, but the PIN tokenization did not
    if (responses.pinResponse.timeout || responses.pinResponse.response == '893' ||
          responses.pinResponse.response == '894') {
          // PIN tokenization timed out or failed because user mistyped their PIN.
          // It makes sense to resubmit PIN only for tokenization
          // Add code here to save responses.panResponse.paypageRegistrationId and load a PIN-only iframe
    }
    else {
        // PIN tokenization returned 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 responses.pinResponse.vantivTxnId and
responses.pinResponse.response received,
        // plus information you have about the consumer to your servers to facilitate debugging, i.e.,
customer ip address, user agent and time
    }
  }
else if (responses.pinResponse.response && responses.pinResponse.response == '870') {
    // The PIN tokenization succeeded, but the PAN tokenization did not
    if (responses.panResponse.timeout || responses.panResponse.response == '871' ||
            responses.panResponse.response == '872' || response.panResponse.response == '873' ||
            response.panResponse.response == '874' || response.panResponse.response == '876') {
        // PAN tokenization timed out or failed because user mistyped their account number.
        // It makes sense to resubmit PAN only for tokenization
        // Add code here to save responses.pinResponse.pinCheckoutId and load a PAN-only iframe
    }
    else {
        // PAN tokenization returned 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 responses.panResponse.vantivTxnId and
responses.panResponse.response received,
        // plus information you have about the consumer to your servers to facilitate debugging, i.e.,
customer ip address, user agent and time
    }
  }
else if (responses.pinResponse.response && (responses.pinResponse.response == '893' ||
responses.pinResponse.response == '894')
        && responses.panResponse.response && (responses.panResponse.response == '871'
        || responses.panResponse.response == '872' || responses.panResponse.response == '873'
        || responses.panResponse.response == '873' || responses.panResponse.response == '874'
        || responses.panResponse.response == '876')) {
    // Both PAN and PIN tokenization failed because of user mistakes. No need to load a different iframe.
    alert("Please check and re-enter your account number and PIN and try again");
  }
    else {
```

```
        // Both PAN and PIN tokenizations returned non-recoverable or unknown error codes.
        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 values received,
        // plus information you have about the consumer to your servers to facilitate debugging, i.e., customer
ip address, user agent and time
      }
    }
}
```

2.2.9.2 Handling Errors - iFrame Version 3

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 FIS-Worldpay Implementation to provide error styles. The codes correspond to the response codes outlined in eProtect-Specific Response Codes on page 13.

- 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** and **PIN** fields, these classes are added to the `div` in the iFrame with the existing class `cvvDiv` or `pinDiv`:
  - `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;
}
```

2.2.9.3 Handling Errors - iFrame Version 4

In case of errors in the iFrame—and for the iFrame to be accessible—Version 4 iFrame uses JavaScript to add an error message to the `div` containing the field with the error.

> **NOTE:** The error handling mechanism described for Version 3 above continues to operate in iFrame Version 4 and can be used to add styling to fields containing errors. However, using the `::before` and `::after` CSS pseudo-elements to add error messages (as shown in the code example for Version 3 above) is not recommended. Text added using this method cannot be detected by assistive technologies, e.g., screen readers.

Use the `customErrorMessages` property of the `config` object in the iFrame to specify a custom error message for input errors. The value of the `customErrorMessages` property is an object whose keys are the error codes listed in Table 1-3, "eProtect-Specific Response Codes Received in Browsers or Mobile Devices". The values are the error messages displayed when the corresponding error is detected.

You can specify separate error messages using the keys **886-month** and **886-year** for invalid expiration month and invalid expiration year. eProtect can display the expiration year error (key **886**) when either the expiration month error, expiration year error or both are encountered.

If a custom error message is not provided for an error code, the default error message displays (as listed in Table 2-5). Use `null` to display no error message for a specific error code.

Error Message Clearing

In iFrame Version 4, error messages and CSS error classes are not cleared when the input field with the error receives the focus event (as happens in version 3 for CSS error classes). Errors in Version 4 are cleared when:

- The form is submitted.
- A field passes an in-line field validation, if the in-line field validation feature is enabled (by setting the `enhancedUxFeatures.inlineFieldValidations` property to `true`).

This makes the error messages available to customers who use screen readers when they return to a field to fix an error.

> **NOTE:** Specifying `null` for an error code suppresses the error message for that error, however the CSS classes described above continue to be used.

```
var
  configure
  = {
    "paypageId":document.getElementById("request$paypageId").value,
    ,
    …
    "customErrorMessages": {
      "871":"Not enough digits in card num",
    },
    …
};
```

TABLE 2-5 Default Error Messages

`customErrorMessages` **Key**	**Default Error Message**	**Notes**
871	Invalid account number	Triggered when card number fails mod10 check.
872	Account number too short	
873	Account number too long	
874	Account number not numeric	
876	Invalid account number	
881	Card validation number not numeric	
882	Card validation number too short	
883	Card validation number too long	
886-month	Expiration month invalid	
886-year	Expiration year invalid	
886	Expiration date invalid	Triggered when either the expiration moth, expiration year, or both are invalid.
893	PIN too short	
894	PIN too long	

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 Worldpay and receive a Registration ID in the response. This section also provides information on the following payment methods:

- Using the Worldpay Mobile API for Apple Pay
- Using the Worldpay Mobile API for Visa Checkout
- Using the Worldpay Mobile API for Google 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 2-6. 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 2-6 POST Headers, Parameters, and URL

Component	Element	Description
Headers (optional)	`Content-Type: application/x-www-form-urlencoded`	
	`Host:` **`request.eprotect.vantivprelive.com`**	
	`User-Agent = "User-Agent" ":" 1*( product | comment )` *For example:* `User-Agent: Vantiv/1.0 CFNetwork/459 Darwin/10.0.0.d3`	
Parameters (required)	`paypageId`	The unique number assigned by Implementation.
	`reportGroup`	(*Not used by eProtect integrations on the Core platform.*) A unique value that you assign.
	`orderId`	A unique value that you assign (string, max length: 25 char.). See full definition on page 33.
	`id`	A unique value that you assign (string, max length: 25 char.). See full definition on page 33.
	`accountNumber`	A unique value that you assign. (*Not used in Apple Pay transactions.*)

TABLE 2-6 POST Headers, Parameters, and URL (Continued)

Component	Element	Description
Parameters (optional)	`pciNonSensitive`	Bypasses 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. A value of `false` allows certain methods of payment other private label cards to make use of the MOD10 check and return the BIN (first 6 digits of card number). When you set the `pciNonSensitive` parameter to `false`, the type attribute is not returned (because the method of payment cannot be determined) and does not cause a validation failure. See Notes on the PCI Non-Sensitive Value Feature, next. ***Note***: If you use this parameter with a value of `true`, the card type and BIN are not returned in the response.
	`cvv`	The card validation number, either the CVV2 (Visa), CVC2 (Mastercard), or CID (American Express and Discover) value.
URL	**`https://request.eprotect.vantivprelive.com/eProtect/paypage`**	

NOTE: The URL in this example script (in red) should only be used in the certification and testing environment. Before using your checkout page with 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" -H "User-Agent: Vantiv/1.0 CFNetwork/459 Darwin/10.0.0.d3"
-d"paypageId=a2y4o6m8k0&
reportGroup=*merchant1500&orderId=PValid&id=12345&accountNumber=ACCOUNT_NUMBER&cvv=CVV
"https://request.eprotect.vantivprelive.com/eProtect/paypage
```

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","accountRangeId":"276989386848","paypageReg
istrationId":"amNDNkpWckVGNFJoRmdNeXJUOHl4Skh1TTQ1Z0t6WE9TYmdqdjBJT0F5N28zbUpxdlhGazZFdmlCSzdTN3p
tKw\u003d\u003d","type":"VI","id":"12345","vantivTxnId":"83088059521107596","message":"Success","
orderId":"PValid","reportGroup":"*merchant1500","response":"870","responseTime":"2014-02-07T17:04
:04"}
```

2.3.1.3 Sample Response - Method of Payment not Identified

The response received when eProtect cannot determine the method of payment and the request uses a `pciNonsensitive` value of `false` is similar to the following:

```
jQuery172022762244707120605_1616695763529({"paypageRegistrationId":"0000009999999990000","bin":"0
00000","targetServer":"primary","vantivTxnId":"82930559381342490","orderId":"order_123","response
":"870","responseTime":"2021-03-25T18:09:52","message":"Success","reportGroup":"*merchant1500","i
d":"987012"})
```

Table 2-7 lists the parameters included in the response.

TABLE 2-7 Parameters Returned in POST Response

Parameter	Description
`bin`	The bank identification number (BIN), which is the first six digits of the credit card number
`firstSix`	(Mirrored back from the request) The first six digits of the credit card number.
`lastFour`	(Mirrored back from the request) The last four digits of the credit card number.
`accountRangeId`	The Worldpay-assigned value representing the account range of the card. (*Only seen by merchants enabled for Issuer Insights, an FIS-Worldpay Value-added Service.*) This value can be used to correlate various data points across card types and issuers. The account range ID is tied to the Issuer Insights Scheduled Secure Report (SSR). See the *Worldpay eComm Scheduled Secure Reports Reference Guide* for more information on the Issuer Insights report.
`paypageRegistrationId`	The temporary identifier used to facilitate the mapping of a token to a card number.
`type`	The method of payment for this transaction (VI=Visa, MC=Mastercard, AX=Amex, DI=Discover). Not returned when the method of payment cannot be determined.
`id`	(Mirrored back from the request) The merchant-assigned unique value representing this transaction in your system. **Type**: String **Max Length**: 25 characters
`vantivTxnId`	The automatically-assigned unique transaction identifier.
`message`	The transaction response returned by Worldpay, corresponding to the `response` reason code. If the transaction was declined, this message provides a reason.

TABLE 2-7 Parameters Returned in POST Response (Continued)

Parameter	Description
`orderId`	(Mirrored back from the request) The merchant-assigned unique value representing the order in your system. **Type**: String **Max Length**: 25 characters
`reportGroup`	(Mirrored back from the request) The cnpAPI required attribute that defines under which merchant sub-group this transaction will be displayed in eCommerce iQ Reporting and Analytics.
`response`	The three-digit transaction response code returned by Worldpay for this transaction.
`responseTime`	The date and time (GMT) the transaction was processed.

2.3.2 Using the Worldpay Mobile API for Apple Pay

In this scenario, your native iOS application performs an HTTPS POST of the Apple Pay PKPaymentToken using the Worldpay 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/documentation/passkit/pkpaymenttoken) 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 63) 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 and submits an Authorization/Sale transaction to your FIS-Worldpay payment API using the Registration ID.

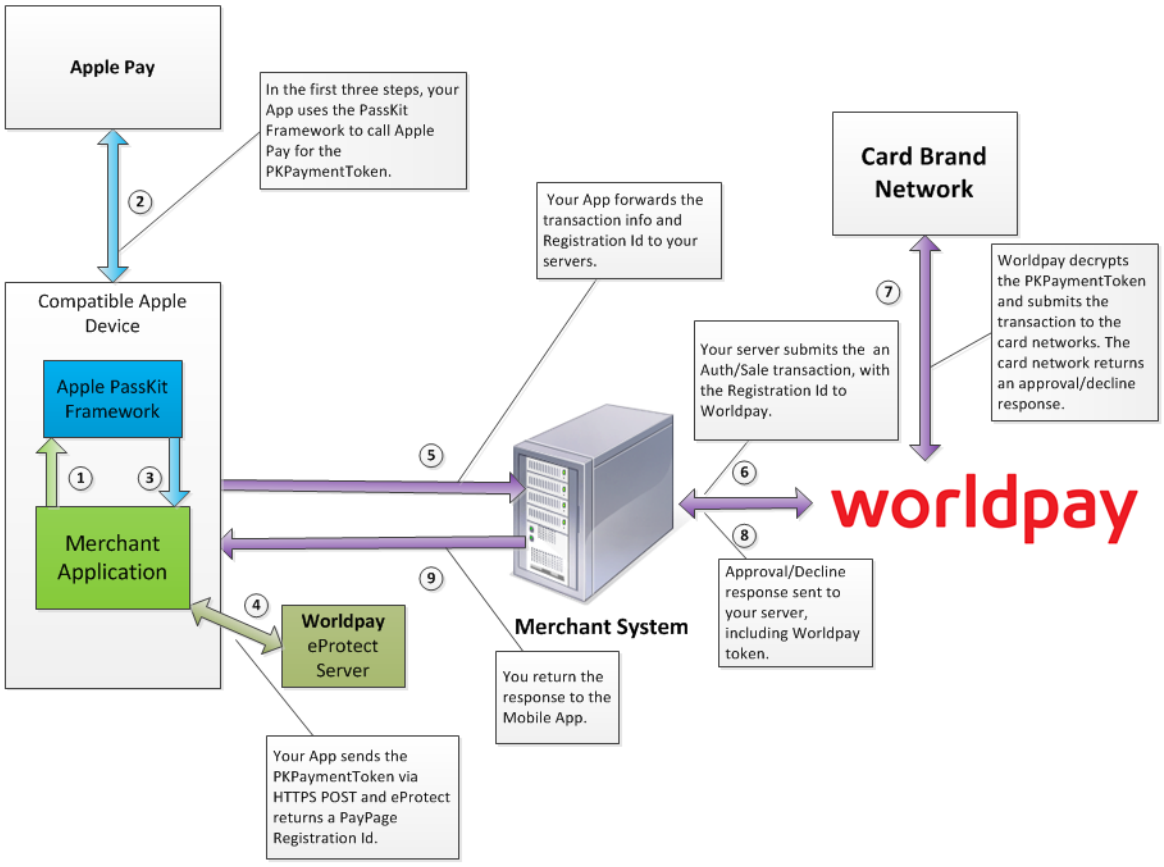
> **NOTE:** In the FIS-Worldpay payment API, it is not necessary to set the date in the Authorization/Sale

7. Using the private key, Worldpay decrypts the PKPaymentToken associated with the Registration ID and submits the transaction with the appropriate information to the card networks for approval.

8. Worldpay sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response and includes the Worldpay token.
9. You return the Approval/Decline message to your mobile application.

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

FIGURE 2-3 Data/Transaction Flow using the Worldpay 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 58, using the components listed in the Table 2-6 as well as those listed in Table 2-8 (all required). See the Sample Apple Pay POST Request and Sample Apple Pay POST Response below.

```
var applepay = {};
  applepay.data = "",
  applepay.signature = "";
  applepay.version = "";
  applepay.header = {};
  applepay.header.applicationData = "";
  applepay.header.ephemeralPublicKey = "";
  applepay.header.publicKeyHash = "";
  applepay.header.transactionId = "";
```

Table 2-8 describes these components.

TABLE 2-8 Worldpay 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:

> **NOTE:** The following fields from the `PKPaymentToken` must be URL-encoded: `data`, `ephemeralPublicKey`, `publicKeyHash`, and `signature`.

```
curl --verbose -H "Content-Type: application/x-www-form-urlencoded" -H "request.eprotect.
vantivprelive.com" -H "User-Agent:Vantiv/1.0 CFNetwork/459 Darwin/10.0.0.d3" -d"paypageId=a2y4o6m8k
0&reportGroup=*merchant1500&orderId=PValid&id=1234&applepay.data=HT897mACd%2F%2FTpWe10A5y9RmL5UfboT
iDIvjni3zWFtyy8dtv72RJL1bk%2FU4dTDlrq1T1V2l0TSnI%0APLdOnnHBO51bt9Ztj9odDTQ5LD%2F4hMZTQj3lBRvFOtTtjp
9ysBAsydgjEjcCcbnkx7dCqgnwgzuz%0Ay7bX%2B5Fo8a8RKqoprkDPwIMWOC9yWe7MQw%2FboM5NY2QtIcIvzbLFcYUxndYTg0
IXNBHNzsvUOjmw%0AvEnMhXxeCH%2BC4KoC6MEsAGK5rH1T5dSvTZzHF5c12dpsqdI73%2FBk6qEcdlT7gJKVmyDQC%2FNFxJ0X
%0AF993Of6ejQDJq6BZsz8X7kYCyJdI%2FPFJPZp4e3L%2FtCsBDUTJAgFLt2xF8HWaPoW8psILOGCCvJQm%0ATR1m7ODtSChaW
Ob7eYm1BpNiD3wkCH8nmIMrlnt3KP4SeQ%3D%3D&applepay.signature=MIAGCSqGSIb3DQEHAqCAMIACAQExDzANBglghkgB
ZQMEAgEFADCABgkqhkiG9w0BBwEAAKCAMIICvzCCAmWgAwIBAgIIQpCV6UIIb4owCgYIKoZIzj0EAwIwejEuMCwGA1UEAwwlQXB
wbGUgQXBwbGljYXRpb24gSW50ZWdyYXRpb24gQ0EgLSBHMzEmMCQGA1UECwwdQXBwbGUgQ2VydGlmaWNhdGlvbiBBdXRob3JpdH
kxEzARBgNVBAoMCkFwcGxlIEluYy4xCzAJBgNVBAYTAlVTMB4XDTE0MDUwODAxMjMzOVoXDTE5MDUwNzAxMjMzOVowXzElMCMGA
1UEAwwcZWNjLXNtcC1icm9rZXItc2lnbl9VQzQtUFJPRDEUMBIGA1UECwwLaU9TIFN5c3RlbXMxEzARBgNVBAoMCkFwcGxlIElu
Yy4xCzAJBgNVBAYTAlVTMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEwhV37evWx7Ihj2jdcJChIY3HsL1vLCg9hGCV2Ur0pUE
bg0IO2BHzQH6DMx8cVMP36zIg1rrV1O%2F0komJPnwPE6OB7zCB7DBFBggrBgEFBQcBAQQ5MDcwNQYIKwYBBQUHMAGGKWh0dHA6
Ly9vY3NwLmFwcGxlLmNvbS9vY3NwMDQtYXBwbGVhaWNhMzAxMB0GA1UdDgQWBBSUV9tv1XSBhomJdi9%2BV4UH55tYJDAMBgNVH
RMBAf8EAjAAMB8GA1UdIwQYMBaAFCPyScRPk%2BTvJ%2BbE9ihsP6K7%2FS5LMDQGA1UdHwQtMCswKaAnoCWGI2h0dHA6Ly9jcm
wuYXBwbGUuY29tL2FwcGxlYWljYTMuY3JsMA4GA1UdDwEB%2FwQEAwIHgDAPBgkqhkiG92NkBh0EAgUAMAoGCCqGSM49BAMCA0g
AMEUCIQCFGdtAk%2B7wXrBV7jTwzCBLE%2BOcrVL15hjif0reLJiPGgIgXGHYYeXwrn02Zwcl5TT1W8rIqK0QuIvOnO1THCbkhV
owggLuMIICdaADAgECAghJbS%2B%2FOpjalzAKBggqhkjOPQQDAjBnMRswGQYDVQQDDBJBcHBsZSBSb290IENBIC0gRzMxJjAkB
gNVBAsMHUFwcGxlIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MRMwEQYDVQQKDApBcHBsZSBJbmMuMQswCQYDVQQGEwJVUzAeFw0x
NDA1MDYyMzQ2MzBaFw0yOTA1MDYyMzQ2MzBaMHoxLjAsBgNVBAMMJUFwcGxlIEFwcGxpY2F0aW9uIEludGVncmF0aW9uIENBIC0
gRzMxJjAkBgNVBAsMHUFwcGxlIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MRMwEQYDVQQKDApBcHBsZSBJbmMuMQswCQYDVQQGEw
JVUzBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABPAXEYQZ12SF1RpeJYEHduiAou%2Fee65N4I38S5PhM1bVZls1riLQl3YNIk5
7ugj9dhfOiMt2u2ZwvsjoKYT%2FVEWjgfcwgfQwRgYIKwYBBQUHAQEEOjA4MDYGCCsGAQUFBzABhipodHRwOi8vb2NzcC5hcHBs
ZS5jb20vb2NzcDA0LWFwcGxlcm9vdGNhZzMwHQYDVR0OBBYEFCPyScRPk%2BTvJ%2BbE9ihsP6K7%2FS5LMA8GA1UdEwEB%2FwQ
FMAMBAf8wHwYDVR0jBBgwFoAUu7DeoVgziJqkipnevr3rr9rLJKswNwYDVR0fBDAwLjAsoCqgKIYmaHR0cDovL2NybC5hcHBsZS
5jb20vYXBwbGVyb290Y2FnMy5jcmwwDgYDVR0PAQH%2FBAQDAgEGMBAGCiqGSIb3Y2QGAg4EAgUAMAoGCCqGSM49BAMCA2cAMGQ
CMDrPcoNRFpmxhvs1w1bKYr%2F0F%2B3ZD3VNoo6%2B8ZyBXkK3ifiY95tZn5jVQQ2PnenC%2FgIwMi3VRCGwowV3bF3zODuQZ%
2F0XfCwhbZZPxnJpghJvVPh6fRuZy5sJiSFhBpkPCZIdAAAxggFfMIIBWwIBATCBhjB6MS4wLAYDVQQDDCVBcHBsZSBBcHBsaWN
hdGlvbiBJbnRlZ3JhdGlvbiBDQSAtIEczMSYwJAYDVQQLDB1BcHBsZSBDZXJ0aW9uIEF1dGhvcml0eTETMBEGA1UECg
wKQXBwbGUgSW5jLjELMAkGA1UEBhMCVVMCCEKQlelCCG%2BKMA0GCWCGSAFlAwQCAQUAoGkwGAYJKoZIhvcNAQkDMQsGCSqGSIb
3DQEHATAcBgkqhkiG9w0BCQUxDxcNMTQxMDAzMjE1NjQzWjAvBgkqhkiG9w0BCQQxIgQgg8i4X6yRAU7AXS1lamCf02UIQlpUvN
PToXUaamsFUT8wCgYIKoZIzj0EAwIERzBFAiBe17NGTuuk%2BW901k3Oac4Z90PoMhN1qRqnij9KNEb%2FXAIhALELZyDWw0fQM
8t0pXO86gg9xXFz424rEMlJ01TM1VxhAAAAAAAA&applepay.version=EC_v1&applepay.header.applicationData=4964
61ea64b50527d2d792df7c38f301300085dd463e347453ae72debf6f4d14&applepay.header.ephemeralPublicKey=MFk
wEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEarp8xOhLX9QliUPS9c54i3cqEfrJD37NG75ieNxncOeFLkjCk%2FBn3jVxHlecRwYq
e%2BAWQxZBtDyewaZcmWz5lg%3D%3D&applepay.header.publicKeyHash=zoV5b2%2BmqnMIxU9avTeqWxc7OW3fnKXFxyhY
0cyRixU%3D&applepay.header.transactionId=23e26bd8741fea9e7a4d78a69f4255b315d39ec14233d6f1b32223d199
9fb99f" https://request.eprotect.vantivprelive.com/eProtect/paypage
```

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

2.3.2.3 Sample Apple Pay POST Response

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


```
{"bin":"410000","firstSix":"410000","lastFour":"0001","paypageRegistrationId":"S0ZBUURMTlZkMTgrbW
1IL3BZVFFmaDh0M0hjdDZ5RXcxQzRQUkJRKzdVc3JURXp0N0JBdmhDN05aTllUQU5rY1RCMDhLNXg2clI0cDV3Sk5vQmlPTjY
3V2plbDVac0lqd0FkblYwVTdQWms9","type":"VI","id":"1234","vantivTxnId":"82826626153431509","message
":"Success","orderId":"PValid","reportGroup":"*merchant1500","response":"870","responseTime":"201
5-01-19T18:35:27","expDate":"0718"}
```


2.3.3 Using the Worldpay 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:

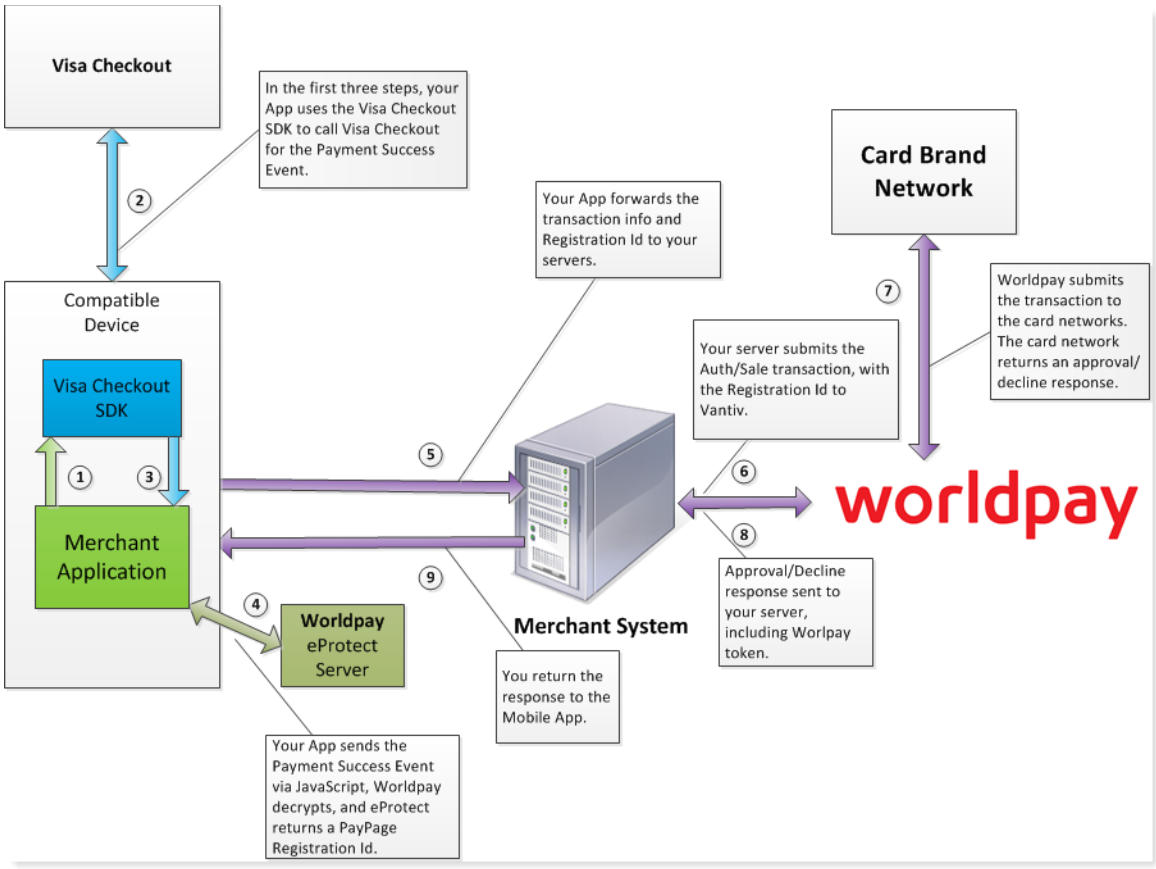
1. When the consumer selects the Visa Checkout option from your application, your application makes use of the Visa Checkout SDK to request payment data from Visa Checkout.
2. When Visa Checkout receives the call from your application, Visa creates a Payment Success event using the Worldpay API key or Encryption 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) to your application.

In this scenario, your native application performs an HTTPS POST of the Visa Checkout SDK using the Worldpay 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 - Worldpay Mobile API for Visa Checkout), Worldpay 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. Worldpay submits the transaction with the appropriate information to the card networks for approval.
8. Worldpay sends the Approval/Decline message back to your system. This message is the standard format for an Authorization or Sale response and includes the Worldpay 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 Worldpay Mobile API for Visa Checkout



2.3.3.1 Sending Worldpay the Required Fields

Construct your HTTPS POST as detailed above using the components listed in Table 2-9 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 2-9 HTTPS POST Required Components - Worldpay Mobile API for Visa Checkout

Parameter Name	Description
`visaCheckout.apiKey`	The API key used to identify the shared secret used by Visa Checkout and Worldpay to decrypt the `encPaymentData` and `encKey` fields. You use both a live key and a sandbox key, which are different from each other.
`visaCheckout.encKey`	Encrypted key used to decrypt `encPaymentData`. Worldpay uses its shared secret identified by the `apiKey` or `encryptionKey` to decrypt this key.
`visaCheckout.encryptionKey`	The Encryption key used to identify the shared secret used by Visa Checkout and Worldpay to decrypt the `encPaymentData` and `encKey` fields. You use both a live key and a sandbox key, which are different from each other. When the `encryptionKey` is not present, Worldpay uses the `apiKey`. ***Note***: The `apiKey` is a mandatory field and must be present whether or not the `encryptionKey` is present.
`visaCheckout.encPaymentData`	Encrypted consumer and payment data that can be used to process the transaction. Worldpay uses the decrypted `encKey` to decrypt this value. The decrypted value is 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=123
45&visaCheckout.apiKey=ENIWGH9WQ4RZMHU9TPKX21EwEluZeS4zp252_mnXTRKuuvUG4&visaCheckout.encrypt
ionKey=5M6PIKT7XURYXV0ZXFMB13RsoW6GlkJCB12FRYGolGw8B3-eM&visaCheckout.encKey=I7Q2OFSekX8PZ239
ZSX8T8AqAZwOtyzTS%2BH%2F9Rj4%2FOXQn2uDTbNIPHJY4KXu5e9UV2Gphi3Mm2DY%2Fb947DaTUO7eD0ijwIUi8wcJQ
Zjddtoe3oC7TLVSqIvtwNb0UjXf&visaCheckout.callid=111111111111111111&visaCheckout.encPaymentDa
ta=YnIJmecBIamQMxbUTVjNXJbxf%2BGarZSqvC8T%2FWRz1AqG1jGHA3TCbu3V268uNIEDKFX4cbgHF6pxSCJjNPMrUe
Mef3Y3%2Fvd08GgSsucrLS4MLPwj4LBtX7G4mt1HFiij4E0ESMedNxywH%2BJ68Oqu%2BTkN%2FwHIJNfAjg2F7pHx3qD
uisCih%2BPkCAWOAjrHxOS1LKf%2FzkerBfR0U0RYZBuxsD4zWJIUfhSbtrbnmVhD%2BIIZSr%2BT%2F6zvPVtInnND3y
qFf%2Bmy7IymV5RUV9Ta41yVuEc&
```

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","la
stFour":"4821","visaCheckoutResponse":"{\"userData\":{\"userFirstName\":\"Jonathon\",\"userLastNa
me\":\"Ross\",\"userFullName\":\"Jonathon Ross\",\"userName\":\"jross@
vantiv.com\",\"encUserId\":\"s/XZjc5uAHsIHCrH+NwOkqfIrWmM1oO4lcj3Tkw/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\":{\"ca
rdBrand\":\"VISA\",\"cardType\":\"CREDIT\"},\"billingAddress\":{\"personName\":\"Jonathon
Ross\",\"firstName\":\"Jonathon\",\"lastName\":\"Ross\",\"line1\":\"900 Chelmsford
Street\",\"line2\":\"Floor 11 co Vantiv eCommerce\",\"city\":\"Lowell\",\"stateProvince
Code\":\"MA\",\"postalCode\":\"01851\",\"countryCode\":\"US\",\"phone\":\"9782756684\",\"default\
":false},\"verificationStatus\":\"VERIFIED\",\"expired\":false,\"cardArts\":{},\"issuerBid\":\"14
\",\"nameOnCard\":\"Jonathon Ross\",\"cardFirstName\":\"Jonathon\",\"card
LastName\":\"Ross\",\"expirationDate\":{\"month\":\"12\",\"year\":\"2020\"}},\"shippingAddress\":
{\"id\":\"lY6VPBAi7ajTAS0XqKizskaC0GuTrYzYOtxiC5URnDY\\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\":fal
se}","cnpTxnId":"82832924191048159","orderId":"TC7976_1_viCheckoutPPPost","response":"870","respo
nseTime":"2017-06-27T19:53:24","message":"Success","reportGroup":"VIReportGroup","id":"12345"}
```


2.3.4 Using the Worldpay Mobile API for Google Pay

This is the recommended and typical method of implementing Google Pay for Mobile Applications on the FIS-Worldpay platform. The steps that follow, along with Figure 2-5, illustrate the high-level flow of messages associated with an Google Pay purchase, when utilizing the Worldpay eProtect service.

1. When the consumer clicks the Google Pay button in your application, the action triggers a `PaymentDataRequest` to Google. This process assumes you have integrated with Google using the method that returns the Worldpay low-value token (`paypageRegistrationId`) from Google following the Full Wallet request. For more information see the Google Tutorial Select the 'Vantiv' Gateway.

   > **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.

   Based on the *Define supported payment card networks* Google process, Google supports passing back both `PAN_ONLY` as well as `CRYPTOGRAM_3DS` values. `PAN_ONLY` indicates a card-not-present keyed transaction and `CRYPTOGRAM_3DS` indicates a wallet transaction.

   If you specify both in your Google Pay instantiation, set `assuranceDetailsRequired` under the `baseCardPaymentMethod`, as shown in the following example:

```
const baseCardPaymentMethod = {
      type: 'CARD',
        parameters: {
            allowedAuthMethods: allowedCardAuthMethods,
            allowedCardNetworks: allowedCardNetworks,
            "assuranceDetailsRequired": true
        }
    };
```

See the Google documentation for more information.

2. Upon confirmation of the order by the consumer, your application initiates a `FullWalletRequest` to Google. The cardholder clicks the Google Pay button initialized in step 1, which prompts the cardholder to choose the card on file stored with Google.
3. After receiving the `FullWalletRequest` from your application, Google submits the card information to Worldpay ePprotect. The ePprotect servers return a low-value token (`paypageRegistrationId`).
4. Google returns the low-value token (`paypageRegistrationId`) to your application under the `tokenizationData.token` field along with the Full Wallet information.

   If you set `assuranceDetailsRequired` in Step 1, note the Google Pay `googleresponse` in the `assuranceDetails.cardHolderAuthenticated` section:

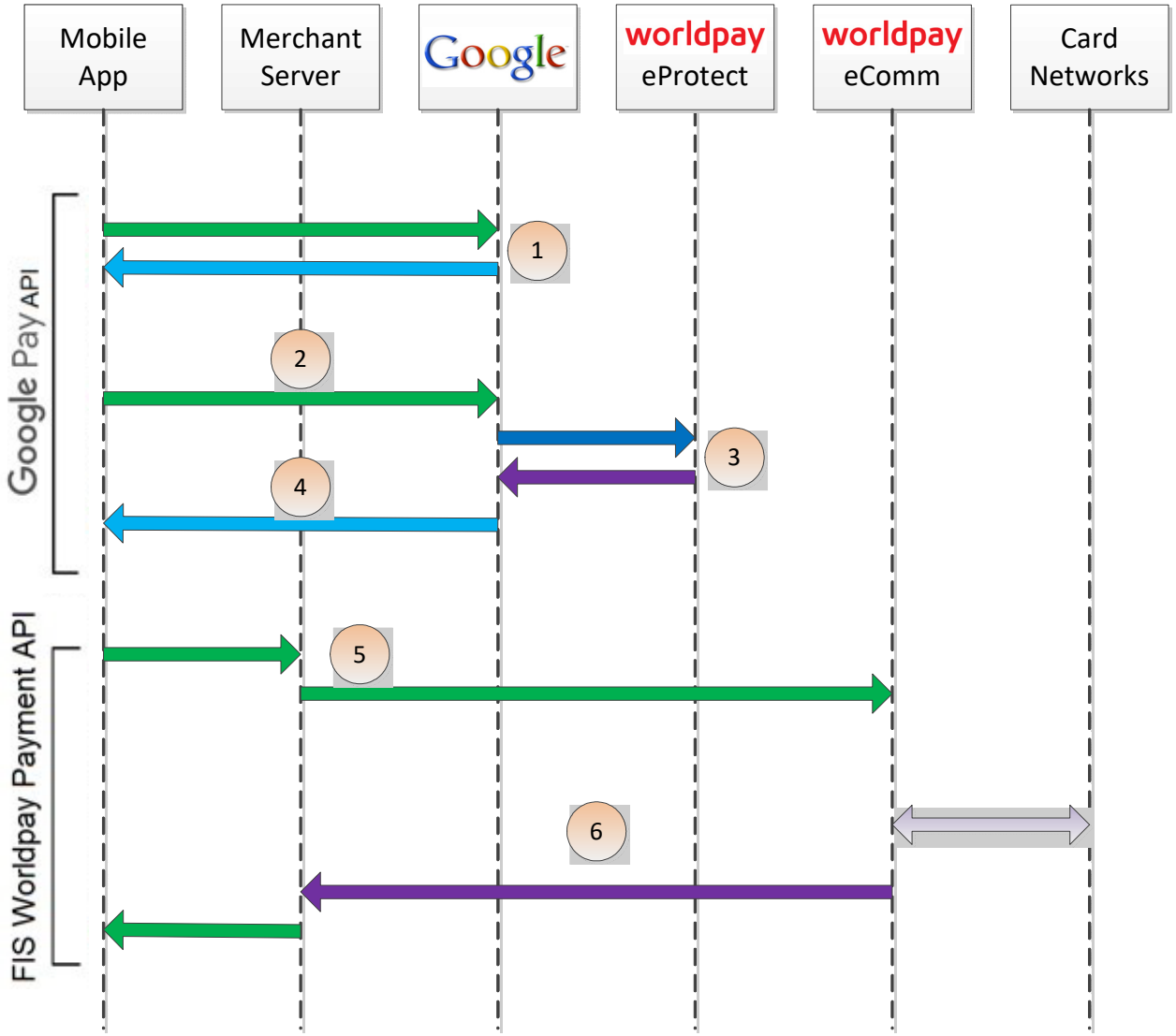
   - A value of **true** indicates that Google passed `CRYPTOGRAM_3DS` to FIS-Worldpay.
   - A value of **false** indicates that Google passed `PAN_ONLY` to FIS-Worldpay.

5. Your servers submit the Auth/Sale transaction to the FIS-Worldpay payment API. Depending on whether Google processed a `PAN_ONLY` or `CRYPTOGRAM_3DS`, ensure that you construct the proper Auth/Sales transaction based on the `cardHolderAuthenticated` value returned by Google.

   > **NOTE:** In the FIS-Worldpay payment API, it is not necessary to set the date in the Authorization/Sale

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

FIGURE 2-5 High Level Message Flow for Google Pay™ using eProtect



2.3.5 Recurring Payments with Apple Pay and Google Pay

When you submit the first transaction in a recurring/installment stream, or when storing credentials for future purchases, you must set the `<processingType>` element to either **initialRecurring**, **initialInstallment**, or **initialCOF** (Card on File), as applicable. With the exception of an American Express transaction, the XML response message includes the `<networkTransactionId>` element. You must retain the value returned for use in future transactions. When you submit the next and all subsequent transactions in the recurring/installment stream, set the `<orderSource>` to recurring or installment as appropriate, and include the `networkTransactionId` value in the `<originalNetworkTransactionId>` element. For a CoF transaction, set the `<orderSource>` to ecommerce and the `<processingType>` element to either **merchantInitiatedCOF**, or **cardholderInitiatedCOF** (Card on File), as applicable.

2.4 Collecting Diagnostic Information

In order to assist Worldpay 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 FIS-Worldpay 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 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 Worldpay 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 112 for definitions of the eProtect-related elements used in these examples.

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

2.5.1 Transaction Types and Examples

This section contains examples of the following transaction types:

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

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

2.5.2 Authorization Transactions

The Authorization transaction enables you to confirm that a customer has submitted a valid payment method with their order and has sufficient funds to purchase the goods or services they ordered.

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

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

2.5.2.1 Authorization Request Structure

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

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

Example: Online Authorization Request

```
<cnpOnlineRequest version="12.23" xmlns="http://www.vantivcnp.com/schema"
 merchantId="100">
 <authentication>
   <user>User Name</user>
   <password>Password</password>
 </authentication>
 <authorization id="834262" reportGroup="ABC Division" customerId="038945">
   <orderId>65347567</orderId>
   <amount>40000</amount>
   <orderSource>ecommerce</orderSource>
   <billToAddress>
     <name>John Smith</name>
```

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

2.5.2.2 Authorization Response Structure

An Authorization response has the following structure:

```
<authorizationResponse id="Authorization Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <cnpTxnId>Transaction Id</cnpTxnId>
  <orderId>Order Id</orderId>
  <response>Response Code</response>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date transaction posted</postDate> (Online Only)
  <message>Response Message</message>
  <authCode>Approval Code</authCode>
  <accountInformation>
  <fraudResult>
  <tokenResponse>
</authorizationResponse>
```

Example: **Online Authorization Response**

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

```
<cnpOnlineResponse version="12.23" xmlns="http://www.vantivcnp.com/schema"
  response="0" message="Valid Format">
  <authorizationResponse id="834262" reportGroup="ABC Division"
  customerId="038945">
    <cnpTxnId>969506</cnpTxnId>
    <orderId>65347567</orderId>
    <response>000</response>
    <responseTime>2009-07-25T15:13:43</responseTime>
    <postDate>2009-07-25</postDate>
    <message>Approved</message>
    <authCode>123457</authCode>
    <fraudResult>
      <avsResult>11</avsResult>
      <cardValidationResult>P</cardValidationResult>
    </fraudResult>
    <tokenResponse>
      <cnpToken>1111000100090005</cnpToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </authorizationResponse>
</cnpOnlineResponse>
```

2.5.3 Sale Transactions

The Sale transaction enables you to both authorize fund availability and deposit those funds by means of a single transaction. The Sale transaction is also known as a conditional deposit, because the deposit takes place only if the authorization succeeds. If the authorization is declined, the deposit will not be processed.

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

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

2.5.3.1 Sale Request Structure

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

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

Example: Online Sale Request

```
<cnpOnlineRequest version="12.23" xmlns="http://www.vantivcnp.com/schema"
 merchantId="100">
 <authentication>
   <user>User Name</user>
   <password>Password</password>
 </authentication>
 <sale id="834262" reportGroup="ABC Division" customerId="038945">
   <orderId>65347567</orderId>
   <amount>40000</amount>
   <orderSource>ecommerce</orderSource>
```

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

2.5.3.2 Sale Response Structure

A Sale response has the following structure:

```
<SaleResponse id="Authorization Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <cnpTxnId>Transaction Id</cnpTxnId>
  <response>Response Code</response>
  <orderId>Order Id</orderId>
  <responseTime>Date and Time in GMT</responseTime>
  <postDate>Date transaction posted</postDate> (Online Only)
  <message>Response Message</message>
  <authCode>Approval Code</authCode>
  <accountInformation>
  <fraudResult>
  <tokenResponse>
</SaleResponse>
```

Example: **Online Sale Response**

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

```
<cnpOnlineResponse version="12.23" 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>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>
  </saleResponse>
</cnpOnlineResponse>
```

2.5.4 Register Token Transactions

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

2.5.4.1 Register Token Request

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

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

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

For PayPage Registration IDs:

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

For Credit Card Register Token request structures, see the *Worldpay eComm cnpAPI Reference Guide*.

> **NOTE:** If you are using OmniTokens, the `<paypageRegistrationId>` value returned by eProtect is numeric only; otherwise the value is alphanumeric.

Example: Online Register Token Request - eProtect

```
<cnpOnlineRequest version="12.23" 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>3854058282021647186</paypageRegistrationId>
    </registerTokenRequest>
  </cnpOnlineRequest>
```

2.5.4.2 Register Token Response

There is no structural difference an Online and Batch response; however, some child elements change depending upon whether the token is for a credit card account, or PayPage registration Id. The response for the will have one of the following structures.

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

```
<registerTokenResponse id="99999" reportGroup="RG1">
  <cnpTxnId>Transaction ID</cnpTxnId>
  <cnpToken>Token</cnpToken>
  <bin>BIN</bin>
  <type>Method of Payment</type>
  <response>Response Code</response>
  <responseTime>Response Time</responseTime>
  <message>Response Message</message>
  <location>Processing Platform Location</location>
</registerTokenResponse>
```

Example: Online Register Token Response - eProtect

```
<cnpOnlineResponse version="12.23" xmlns="http://www.vantivcnp.com/schema"
  id="123" response="0" message="Valid Format" cnpSessionId="987654321">
    <registerTokenResponse id="99999" reportGroup="RG1">
      <cnpTxnId>21122700</cnpTxnId>
      <cnpToken>1111000100360002</cnpToken>
      <bin>400510</bin>
      <type>VI</type>
      <response>801</response>
      <responseTime>2010-10-26T17:21:51</responseTime>
      <message>Account number was successfully registered</message>
    </registerTokenResponse>
</cnpOnlineResponse>
```

2.5.5 Force Capture Transactions

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

CAUTION: Merchants must be authorized by Worldpay 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, Worldpay does not store expiration dates. Therefore, you must always submit an expiration date value with each eProtect cnpAPI transaction.

2.5.5.1 Force Capture Request

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

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

Example: On-Line Force Capture Request

```
<cnpOnlineRequest version="12.23" xmlns="http://www.vantivcnp.com/schema"
 merchantId="100">
 <authentication>
   <user>User Name</user>
   <password>Password</password>
 </authentication>
 <forceCapture  id="834262" reportGroup="ABC Division" customerId="038945">
   <orderId>65347567</orderId>
   <amount>40000</amount>
   <orderSource>ecommerce</orderSource>
   <billToAddress>
     <name>John Smith</name>
     <addressLine1>100 Main St</addressLine1>
     <city>Boston</city>
     <state>MA</state>
     <zip>12345</zip>
     <country>USA</country>
     <email>jsmith@someaddress.com</email>
     <phone>555-123-4567</phone>
   </billToAddress>
```

```
    <paypage>
      <paypageRegistrationId>cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NNlYE4ZW5UTko4NU
  9KK3p1L1p1VzE4ZWVPQVlSUHNITG1JN2I0NzlyTg=</paypageRegistrationId>
      <expDate>1012</expDate>
      <cardValidationNum>712</cardValidationNum>
    </paypage>
  </forceCapture>
</cnpOnlineRequest>
```

2.5.5.2 Force Capture Response

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

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

Example: Force Capture Response

```
<cnpOnlineResponse version="12.23" xmlns="http://www.vantivcnp.com/schema"
  response="0" message="Valid Format">
  <forceCaptureResponse id="2" reportGroup="ABC Division"
   customerId="038945">
    <cnpTxnId>1100030204</cnpTxnId>
    <response>000</response>
    <responseTime>2009-07-11T14:48:48</responseTime>
    <postDate>2009-07-11</postDate>
    <message>Approved</message>
    <tokenResponse>
      <cnpToken>1111000100090005</cnpToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </forceCaptureResponse>
</cnpOnlineResponse>
```

2.5.6 Capture Given Auth Transactions

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

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

2.5.6.1 Capture Given Auth Request

```
<captureGivenAuth id="Capture Given Auth Id" reportGroup="UI Report Group"
customerId="Customer Id">
  <orderId>Order Id</orderId>
  <authInformation>
  <amount>Authorization Amount</amount>
  <orderSource>Order Entry Source</orderSource>
  <billToAddress>
  <shipToAddress>
  <paypage>
```

```
      <paypageRegistrationId>Registation ID returned</paypageRegistrationId>
      <expDate>Card Expiration Date</expDate>
      <cardValidationNum>Card Validation Number</cardValidationNum>
    </paypage>
  </captureGivenAuth>
```

Example: Online Capture Given Auth Request

```
<cnpOnlineRequest version="12.23" 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>2011-06-22</authDate>
   <authCode>111111</authCode>
  </authInformation>
  <amount>40000</amount>
  <orderSource>ecommerce</orderSource>
  <billToAddress>
   <name>John Smith</name>
   <addressLine1>100 Main St</addressLine1>
   <city>Boston</city>
   <state>MA</state>
   <zip>12345</zip>
   <country>USA</country>
   <email>jsmith@someaddress.com</email>
   <phone>555-123-4567</phone>
  </billToAddress>
  <paypage>
   <paypageRegistrationId>cDZJcmd1VjNlYXNaSlRMTGpocVZQY1NNlYE4ZW5UTko4NU
9KK3p1L1p1VzE4ZWVPQVlSUHNITG1JN2I0NzlyTg=</paypageRegistrationId>
   <expDate>1012</expDate>
   <cardValidationNum>000</cardValidationNum>
  </paypage>
 </captureGivenAuth>
</cnpOnlineRequest>
```

2.5.6.2 Capture Given Auth Response

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

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

Example: Online Capture Given Auth Response

```
<cnpOnlineResponse version="12.23" xmlns="http://www.vantivcpn.com/schema"
 response="0" message="Valid Format">
 <captureGivenAuthResponse id="2" reportGroup="ABC Division"
  customerId="038945">
  <cnpTxnId>1100030204</cnpTxnId>
  <response>000</response>
  <responseTime>2011-07-11T14:48:48</responseTime>
  <postDate>2011-07-11</postDate>
  <message>Approved</message>
  <tokenResponse>
    <cnpToken>1111000100090005</cnpToken>
    <tokenResponseCode>801</tokenResponseCode>
    <tokenMessage>Account number was successfully registered</tokenMessage>
    <type>VI</type>
    <bin>402410</bin>
  </tokenResponse>
 </captureGivenAuthResponse>
</cnpOnlineResponse>
```

2.5.7 Credit Transactions

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

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

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

2.5.7.1 Credit Request Transaction

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

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

Example: Online Credit Request Transaction

```
<cnpOnlineRequest version="12.23" xmlns="http://www.vantivcnp.com/schema"
  merchantId="100">
  <authentication>
    <user>User Name</user>
    <password>Password</password>
  </authentication>
  <credit id="834262" reportGroup="ABC Division" customerId="038945">
    <orderId>65347567</orderId>
    <amount>40000</amount>
```

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

2.5.7.2 Credit Response

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

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

Example: Online Credit Response

```
<cnpOnlineResponse version="12.23" xmlns="http://www.vantivcnp.com/schema"
  response="0" message="Valid Format">
  <creditResponse customerId="038945" id="5" reportGroup="ABC Division">
    <cnpTxnId>1100030204</cnpTxnId>
    <response>001</response>
    <responseTime>2009-08-11T14:48:48</responseTime>
    <postDate>2009-08-11</postDate>
    <message>Transaction received</message>
    <tokenResponse>
```

```
      <cnpToken>1111000100090005</cnpToken>
      <tokenResponseCode>801</tokenResponseCode>
      <tokenMessage>Account number was successfully registered</tokenMessage>
      <type>VI</type>
      <bin>402410</bin>
    </tokenResponse>
  </creditResponse>
</cnpOnlineResponse>
```

2.6 Testing and Certification

The FIS-Worldpay payment API 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 Worldpay Conversion Manager. This section provides the specific data you must use in your eProtect transactions when performing the required tests. Use of this data allows the validation of your transaction structure/syntax, as well as the return of a response file containing known data.

The testing process for eProtect includes browser and/or mobile native application interaction, JavaScript interaction, and transaction requests as well as cnpAPI responses with the Registration ID.

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

See Certification and Testing Environments on page 11 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 the FIS-Worldpay payment API 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 13 for definitions of the response codes.

2.6.1 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 28.
   - Integrating iFrame into your Checkout Page on page 42.
   - Integrating eProtect Into Your Mobile Application on page 58.
2. Verify that you are using the appropriate URL (see Table 1-2, "eProtect Certification, Testing, and Production URLs" on page 12) 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** fromTable 2-10. When performing these tests, you can use any expiration date and card type.
4. Verify that your results match the **Result** column in Table 2-10.

TABLE 2-10 Expected eProtect Test Results

Test Case	Card Number	Card Validation Number	Response Code	Result
NOTE: Card Numbers are split into two parts; join **Part 1** and **Part 2** to obtain actual number to use.				
1	Part 1: 51120100 Part 2: 00000003	Any 3-digit	870 (Success)	Registration ID is generated and the card is scrubbed before the form is submitted.
2	Part 1: 445701000 Part 2: 00000009	Any 3-digit	871	Checkout form displays error message to cardholder, for example, “Invalid Card Number - Check and retry (not Mod10).” Not applicable when the PCI non-sensitive parameter is set to ***true***.
3	Part 1: 44570100000 Part 2: 0000000006	Any 3-digit	873	Checkout form displays error message to cardholder, for example, “Invalid Card Number - Check and retry (too long).” Not applicable when the PCI non-sensitive parameter is set to ***true***. ***Note***: Do not use when testing iFrame.

TABLE 2-10 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 cardholder, for example, “Invalid Card Number - Check and retry (too short).” Not applicable when the PCI non-sensitive parameter is set to ***true***.
5	Part 1: 44570100 Part 2: B00000006	Any 3-digit	874	Checkout form displays error message to cardholder, 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 cardholder, 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 cardholder, 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 cardholder, 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 cardholder, 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 cardholder, 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 cardholder, for example “Invalid Card Validation Number - Check and retry (too short)”.
12	Part 1: 51120100 Part 2: 00000003	12345	883	Checkout form displays error message to cardholder, for example “Invalid Card Validation Number - Check and retry (too long)”. ***Note***: Do not use when testing iFrame.

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

1. Verify that your cnpAPI template is coded correctly for this transaction type (see Authorization Transactions on page 74).

2. Submit three Authorization transactions using the eProtect data from Table 2-10.
3. Verify that your `authorizationResponse` values match the **Response Code** column.

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

To test the submission of the Registration ID to the FIS-Worldpay payment API 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 Worldpay Conversion Manager or eProtect Implementation Consultant.

worldpay

A

Code Samples and Other Information

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

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

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 Version 3 Hosted iFrame-Integrated Checkout Page
- HTML Example for Version 4 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> </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://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$accountRangeId').value = response.accountRangeId;
          document.getElementById('response$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)");
          }
          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)");
          }
```

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

```
            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'),
        "cvv"   :document.getElementById('cvvNum'),
"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"
            "minPanLength": 16,
            };
          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"/>

    <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="cvvnum" name="cvvnum" size="5"></td></tr>
      <tr><td>Exp Date</td><td><input type="text" id="expDate" name="expDate" size="5"></td></tr>
      <tr><td> </td><td></tr>
      <tr><td></td><td align="right">
      <script>
        document.write('<button type="button" id="submitId" onclick="callEprotect()">Check out
with eProtect</button>');
      </script>
      <noscript>
```

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

```
        <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"/>
    <input type="hidden" id="response$accountRangeId" name="response$accountRangeId"
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 Version 3 Hosted iFrame-Integrated Checkout Page

The HTML code below is an example of a simple checkout page integrated with Version 3 of 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://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>
        </table>
    </form>
  </div>
      <br/>
  <h3>Test Input Fields</h3>
  <table class="testFieldTable">
      <tr>
          <td>Paypage ID</td><td><input type="text" id="request$paypageId"
```

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

```
name="request$paypageId" value="a2y4o6m8k0" disabled/></td>
          <td>Merchant Txn ID</td><td><input type="text" id="request$merchantTxnId"
name="request$merchantTxnId" value="987012"/></td>
      </tr>
      <tr>
          <td>Order ID</td><td><input type="text" id="request$orderId" name="request$orderId"
value="order_123"/></td>
          <td>Report Group</td><td><input type="text" id="request$reportGroup"
name="request$reportGroup" value="*merchant1500" disabled/></td>
      </tr>
      <tr>
          <td>JS Timeout</td><td><input type="text" id="request$timeout"
name="request$timeout" value="15000" disabled/></td>
      </tr>
    </table>
<h3>Test Output Fields</h3>
<table class="testFieldTable">
    <tr>
        <td>Response Code</td><td><input type="text" id="response$code" name="response$code"
readOnly="true"/></td>
        <td>ResponseTime</td><td><input type="text" id="response$responseTime"
name="response$responseTime" readOnly="true"/></td>
    </tr>
    <tr>
        <td>Response Message</td><td colspan="3"><input type="text" id="response$message"
name="response$message" readOnly="true" size="100"/></td>
    </tr>
    <tr><td> </td><td></tr>
    <tr>
        <td>Vantiv Txn ID</td><td><input type="text" id="response$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$expYear" readOnly="true"/></td>
    </tr>
    <tr><td> </td><td></tr>
    <tr>
        <td>First Six</td><td><input type="text"
id="response$firstSix"name="response$firstSix" readOnly="true"/></td>
      <td>Last Four</td><td><input type="text" id="response$lastFour"name="response$lastFour"
readOnly="true"/></td>
      </tr>
      <tr><td>Timeout Message</td><td><input type="text" id="timeoutMessage"
name="timeoutMessage" readOnly="true"/></td></tr>
      <tr><td>Expected Results</td>
          <td colspan="3">
          <textarea id="expectedResults" name="expectedResults" rows="5" cols="100"
readOnly="true">
            CC Num           - Token Generated (with simulator)
            410000&#48;00000001 - 1111222&#50;33330001
            5123456&#55;89012007 - 1112333&#51;44442007
```

```
            3783102&#48;3312332 - 11134444&#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>

function ready(callback) {
  // in case the document is already rendered
  if (document.readyState != 'loading') callback();
  // modern browsers
  else if (document.addEventListener) document.addEventListener('DOMContentLoaded', callback);
  // IE <= 8 for browser's not supporting addEventListener property
  else document.attachEvent('onreadystatechange', function() {
    if (document.readyState == 'complete') callback();
  });
}
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$accountRangeId').value =
response.accountRangeId;
              document.getElementById('response$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$paypageId").value,
    "style":"test",
    "reportGroup":document.getElementById("request$reportGroup").value,
    "timeout":document.getElementById("request$timeout").value,
    "div": "eProtectiframe",
    "callback": eProtectiframeClientCallback,
    "maskAfterSuccessValue": 'Z',
    "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",
      "pin":"PIN Placeholder"
    },
      "inputsEmptyCallback": inputsEmptyCallback,
      "enhancedUxFeatures" : {
        "inlineFieldValidations": true,
        "expDateValidation": false,
        "enhancedUxVersion": 2
        }
    "minPanLength": 16,
    "iFrameTitle":"My Custom Title",
    "label" : {
      "accountNumber":"Account Number",
      "expDate" : "Exp Date",
      "cvv" : "CVV",
      "pin":"Pin"

    },
  };
  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);

  function checkPayframeLoaded(){
      if(iframeIsReady===true){
          //code changes
      }

    };

    checkPayframeLoaded();


  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;
```

```
  };


window.onmessage = function(event) {
  if(event.data === "checkoutWithEnter") {
       //Captures Enter even from iFrame
    var message = {
             "id": document.getElementById("request$merchantTxnId").value,
             "orderId": document.getElementById("request$orderId").value
          };
          startTime = new Date().getTime();
          payframeClient.getCheckoutPin(message);
          return false;
        }
      };

});

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

A.1.4 HTML Example for Version 4 Hosted iFrame-Integrated Checkout Page

The HTML code below is an example of a simple checkout page integrated with **Version 4** of 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>PAN LVT generator</title>
  <style>
      body {
          font-size:10pt;
      }
      .checkout {
          background-color:rgb(255,255,255);
          width: 50%;
      }
      .testFieldTable {
          background-color:lightgrey;
      }
      #submitId {
          font-weight:bold;
          font-size:12pt;
      }
      form#fCheckout {
      }
      iframe {
          height: 50vh;
      }

    </style>

    <script src="https://request.eprotect.vantivprelive.com/eProtect/js/eProtect-iframe-
client4.min.js">
    </script>
</head>
<BODY>
<div class="checkout">
    <h2>Test PAN LVT generator</h2>
    <form method=post id="fCheckout" name="fCheckout" onsubmit="return false;">
        <table id="tCheckout">
            <tr><td colspan="2">
                <div id="payframe">
                </div>
            </td></tr>
            <tr><td colspan="2"><b>***Everything below is Debug information***</b></td></tr>
            <tr><td>Paypage Registration ID</td><td><input type="text" id="paypageRegistrationId"
name="paypageRegistrationId" readOnly="true"/> <--Hidden</td></tr>
            <tr><td>Bin</td><td><input type="text" id="bin" name="bin" readOnly="true"/> <--Hidden</td></tr>
            <tr><td></td><td align="right"><input type="submit" id="submitId"></td></tr>
        </table>
    </form>
  </div>
<br/>
<h3>Test Input Fields</h3>
```

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

```
<table class="testFieldTable">
        <tr>
            <td>Paypage ID</td><td><input type="text" id="request$paypageId" name="request$paypageId"
value="a2y4o6m8k0" disabled/></td>
            <td>Style</td><td><input type="text" id="request$style" name="request$style" value="client4"
disabled/></td>
        </tr>
        <tr>
            <td>Order ID</td><td><input type="text" id="request$orderId" name="request$orderId"
value="order_123"/></td>
            <td>Merchant Txn ID</td><td><input type="text" id="request$merchantTxnId"
name="request$merchantTxnId" value="987012"/></td>
        </tr>
        <tr>
            <td>JS Timeout</td><td><input type="text" id="request$timeout" name="request$timeout"
value="5000" /></td>
            <td>Report Group</td><td><input type="text" id="request$reportGroup" name="request$reportGroup"
value="Cert30 Merchant Rollup**" disabled/></td>
        </tr>
</table>
<h3>Test Output Fields</h3>
<table class="testFieldTable">
        <tr>
            <td>Response Code</td><td><input type="text" id="response$code" name="response$code"
readOnly="true"/></td>
            <td>ResponseTime</td><td><input type="text" id="response$responseTime"
name="response$responseTime" readOnly="true"/></td>
        </tr>
        <tr>
            <td>Response Message</td><td colspan="3"><input type="text" id="response$message"
name="response$message" readOnly="true" size="100"/></td>
        </tr>
        <tr><td> </td><td></tr>
        <tr>
            <td>Vantiv Txn ID</td><td><input type="text" id="response$litleTxnId" name="response$litleTxnId"
readOnly="true"/></td>
            <td>Merchant Txn ID</td><td><input type="text" id="response$merchantTxnId"
name="response$merchantTxnId" readOnly="true"/></td>
        </tr>
        <tr>
            <td>Order ID</td><td><input type="text" id="response$orderId" name="response$orderId"
readOnly="true"/></td>
            <td>Report Group</td><td><input type="text" id="response$reportGroup"
name="response$reportGroup" readOnly="true"/></td>
        </tr>
        <tr><td>Type</td><td><input type="text" id="response$type" name="response$type"
readOnly="true"/></td></tr>
        <tr>
            <td>Expiration Month</td><td><input type="text" id="response$expMonth" name="response$expMonth"
readOnly="true"/></td>
            <td>Expiration Year</td><td><input type="text" id="response$expYear" name="response$expYear"
readOnly="true"/></td>
        </tr>
        <tr><td> </td><td></tr>

        <tr>
            <td>First Six</td><td><input type="text" id="response$firstSix" name="response$firstSix"
readOnly="true"/></td>
            <td>Last Four</td><td><input type="text" id="response$lastFour" name="response$lastFour"
readOnly="true"/></td>
        </tr>

        <tr><td>Timeout Message</td><td><input type="text" id="timeoutMessage" name="timeoutMessage"
readOnly="true"/></td></tr>
```

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

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

function inputsEmptyCallback(res) {
    console.log("inputsEmptyCallback: message received");
    console.log(res);
    var isEmpty = res.allInputsEmpty;
    if (isEmpty) {
        console.log("Card input fields empty");
        $("<p>Inputs are Empty</p>").insertAfter(".checkout");
        return true;
    } else {
        console.log("Card inputs not empty");
        $("<p>Inputs are not Empty</p>").insertAfter(".checkout");
        return false;
    }
}

var configure = {
      "paypageId": document.getElementById("request$paypageId").value,
      "style": document.getElementById("request$style").value,
      "reportGroup": document.getElementById("request$reportGroup").value,
      "timeout": document.getElementById("request$timeout").value,
```

```
        "div": "payframe",
        "callback": payframeClientCallback,
        "showCvv": true,
        "months": {
            "1":"January",
            "2":"February",
            "3":"March",
            "4":"April",
            "5":"May",
            "6":"June",
            "7":"July",
            "8":"August",
            "9":"September",
            "10":"October",
            "11":"November",
            "12":"December"
        },
        "numYears": 8,
        "htmlTimeout": document.getElementById("request$timeout").value,


        "tabIndex": {
            "accountNumber": 1,
            "expMonth": 2,
            "expYear": 3,
            "cvv": 4
        },
        "inputsEmptyCallback": inputsEmptyCallback,
        "clearCvvMaskOnReturn": true,
        "enhancedUxFeatures": {
            "inlineFieldValidations": true,
            "expDateValidation": true
        },
        "customErrorMessages": {
            "872": "Not enough digits in card num"
        }
};

var payframeClient = new EprotectIframeClient(configure);
    //payframeClient.autoAdjustHeight();
    document.getElementById("fCheckout").onsubmit = function(){
        var message = {
              "id":document.getElementById("request$merchantTxnId").value,
              "orderId":document.getElementById("request$orderId").value,
              "pciNonSensitive" : true
        };
        startTime = new Date().getTime();
        payframeClient.getPaypageRegistrationId(message);
        return false;
      };
window.onmessage = function(event) {
  if(event.data === "checkoutWithEnter") {
      //Captures Enter even from iFrame
    var message = {
             "id": document.getElementById("request$merchantTxnId").value,
             "orderId": document.getElementById("request$orderId").value
          };
          startTime = new Date().getTime();
          payframeClient.getCheckoutPin(message);
          return false;
        }
      };

  });
```

```
</script>
</BODY>
</HTML>
```

A.2 Information Sent to Order Processing Systems

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

A.2.1 Information Sent Without Integrating eProtect

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

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

A.2.2 Information Sent with Browser-Based eProtect Integration

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

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

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

response$firstSix - 410000
response$lastFour - 0001
response$accountRangeId - 1234567890123456789

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 73. For a comprehensive list of all cnpAPI elements and usage, see Chapter 4, “cnpAPI Elements” in the *Worldpay eComm 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 <updateCardValidatinNumOnToken> element, which you use to submit either the CVV2 (Visa), CVC2 (Mastercard), or CID (American Express and Discover) value.

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

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

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

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

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

Parent Elements:

card, paypage, token, registerTokenRequest, updateCardValidationNumOnToken, virtualGiftCardResponse

Attributes:

None

Child Elements:

None

A.3.2 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. If you are using OmniTokens, the value is numeric only; otherwise it is alphanumeric.

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. **minLength** = N/A **maxLength** = 25
customerId	String	No	A value assigned by the merchant to identify the consumer. **minLength** = N/A **maxLength** = 50
reportGroup	String	Yes	Required attribute defining the merchant sub-group in eCommerce iQ where this transaction displays. **minLength** = 1 **maxLength** = 25

Child Elements:

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

Optional: orderId, cardValidationNum

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

A.3.7 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 `registerTokenRequest` transaction. **minLength** = N/A **maxLength** = 25
customerId	String	No	The response returns the same value submitted in the `registerTokenRequest` transaction. **minLength** = N/A **maxLength** = 50
reportGroup	String	Yes	The response returns the same value submitted in the `registerTokenRequest` transaction. **minLength** = 1 **maxLength** = 25

Child Elements:

Required: cnpTxnId, response, message, responseTime

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

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, Worldpay 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>
```

worldpay

B

CSS Properties for iFrame API

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

See the section Creating a Customized CSS for iFrame on page 18 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.

CSS Properties not listed - there may be properties not listed in this appendix that you wish to use when creating your style sheet. We do not list every non-allowed CSS property, just those that we explicitly black-list (or that are 'excluded from the white-list'). There may be an opportunity to evaluate new CSS properties to add to the white-list. Please contact your Implementation Consultant to initiate a request for future development consideration of CSS properties.

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

B.1 CSS Property Groups

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

TABLE B-1 Color Properties

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

TABLE B-2 Background and Border Properties

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

TABLE B-2 Background and Border Properties (Continued)

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

TABLE B-2 Background and Border Properties (Continued)

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

TABLE B-3 Basic Box Properties

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

TABLE B-3 Basic Box Properties (Continued)

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

TABLE B-4 Flexible Box Layout

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

TABLE B-4 Flexible Box Layout (Continued)

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

TABLE B-5 Text Properties

Property	Description
hanging-punctuation	Specifies whether a punctuation character may be placed outside the line box
hyphens	Sets how to split words to improve the layout of paragraphs
letter-spacing	Increases or decreases the space between characters in a text
line-break	Specifies how/if to break lines
line-height	Sets the line height
overflow-wrap	Specifies whether or not the browser may break lines within words in order to prevent overflow (when a string is too long to fit its containing box)
tab-size	Specifies the length of the tab-character
text-align	Specifies the horizontal alignment of text
text-align-last	Describes how the last line of a block or a line right before a forced line break is aligned when text-align is “justify”

TABLE B-5 Text Properties (Continued)

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

TABLE B-6 Text Decoration Properties

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

TABLE B-7 Font Properties

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

TABLE B-7 Font Properties (Continued)

Property	Description
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

TABLE B-8 Writing Modes Properties (Continued)

Property	Description
unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
writing-mode	

TABLE B-9 Table Properties

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

TABLE B-10 Lists and Counters Properties

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

TABLE B-11 Animation Properties

Property	Description
@frames	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

TABLE B-11 Animation Properties (Continued)

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

TABLE B-14 Basic User Interface Properties

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

TABLE B-15 Multi-Column Layout Properties

Property	Description
break-after	Specifies the page-, column-, or region-break behavior after the generated box
break-before	Specifies the page-, column-, or region-break behavior before the generated box

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

Property	Description
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
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

TABLE B-21 Speech Properties (Continued)

Property	Description
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

worldpay

C

Sample eProtect Integration Checklist

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

FIGURE C-1 Sample eProtect Integration Checklist

eProtect Integration Checklist

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

Merchant/Organization _____Contact Name_____

Phone_____Date Completed_____

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

____ 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

L

M

N

O

P

R

S

T

V