PayPal™

Website Payments Pro Integration Guide

Last updated: December 2010

PayPal Website Payments Pro Integration Guide

Document Number: 100001.en_US-201012

Contents

Preface

This document describes Website Payments Pro integration.

Intended Audience

This document is intended for merchants and developers implementing Website Payments Pro.

Revision History

Revision history for *PayPal Website Payments Pro Integration Guide*.

TABLE 1.1 Revision history

Date	Description
12/20/10	Replaced deprecated field names in examples.
10/26/10	Added a reference to information on Mobile Express Checkout, which is located in the Epxress Checkout Integration Guide.
08/11/10	Divided the *Website Payments Pro Integration Guide* into 2 guides: the *Website Payments Pro Integration Guide* and the *Express Checkout Advanced Features Guide*.
05/11/10	Added details for integrating parallel payments using the NVP and SOAP API, including use with airlines. Added new Immediate Payment functionality. Updated billing agreements with functionality to obtain the latest billing address, skip billing agreement creation, and clarify use of the BAUpdate API.
03/10/2010	Consolidated all regions (US, UK, and Canada) into one manual. Added additional information about Direct Payment.
10/05/2009	Added Immediate Payment. Edited for technical accuracy. Removed PayPal placement guidelines.
06/30/2009	Added a section on payment review.
06/04/2009	Added a chapter on pre-populating the PayPal review page. Updated PayPal Review pages. Moved some customization topics out of this guide. They are now in the *Merchant Setup and Administration Guide*.

TABLE 1.1 Revision history

Date	Description
04/08/2009	Added a chapter describing the Instant Update Callback API. Added Express Checkout feature of passing AMT=0 to create one or more billing agreements.
12/11/2008	Revised the Website Payments Pro introduction and overview chapters.
11/13/2008	Added information about integrating dynamic images and added information about order details that can be displayed on the PayPal Review page.
06/30/2008	Complete revision.

Where to Go for More Information

- *Express Checkout Advanced Features Guide*
- *Merchant Setup and Administration Guide*

For information on Mobile Express Checkout, see:.

- *Express Checkout Integration Guide*

Documentation Feedback

Help us improve this guide by sending feedback to:

documentationfeedback@paypal.com

1 Introducing Website Payments Pro

You can accept credit and debit cards and PayPal payments directly on your website using 2 API-based solutions: Direct Payment and Express Checkout. You must integrate with both Direct Payment and Express Checkout to use Website Payments Pro.

Getting Related Information

All PayPal documentation is available on x.com, including video demos, forums and developer resources.

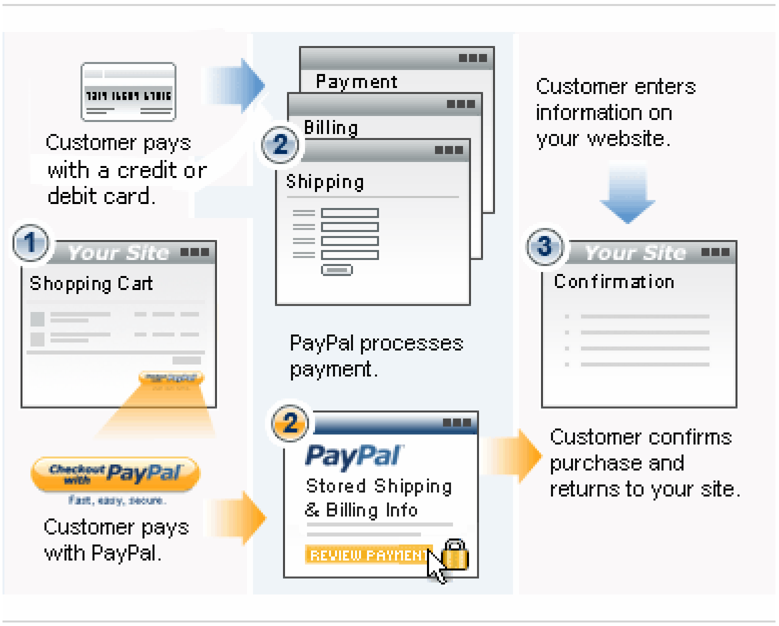
- For information about administrative tasks you can perform from your PayPal account such as adding users, setting up custom page styles, and managing multiple currency balances, see the Merchant Setup and Administration Guide.
- If you use the Payflow API to process transactions with PayPal as your internet merchant account, see Website Payments Pro Payflow Edition Developers Guide.

Website Payments Pro Overview

Website Payments Pro includes Direct Payment, Express Checkout, and additional PayPal solutions and tools, such as Virtual Terminal, Fraud Management Filters, and reference transactions.

- **Direct Payment** enables you to accept both debit and credit cards directly from your site.
- **Express Checkout** enables you to accept payments from PayPal accounts in addition to debit and credit cards.

The following diagram shows the relationship between Direct Payment and Express Checkout to a buyer.



From your shopping cart, a buyer can either checkout with Express Checkout, starting from the **Checkout with PayPal** button on your Shopping Cart page, or pay directly by credit or debit card using Direct Payment.

If a buyer pays using Express Checkout, PayPal provides a checkout experience that streamlines checkout. Even if buyers do not pay using Express Checkout, they can still pay by credit or debit card using Direct Payment. In this case, buyers might need to enter payment, billing, and shipping information. In both cases, buyers stay on your website or are sent to the page of your choice.

You must implement both an Express Checkout flow and a Direct Payment flow to use Website Payments Pro. You implement the Express Checkout flow by calling PayPal's Express Checkout API operations, which guides a buyer through the checkout process. You implement the Direct Payment flow using your own code, for which PayPal provides an API operation to process the credit or debit card payment.

NOTE: Purchases through Direct Payment are not covered by the PayPal Seller Protection Policy.

Additional Features of Website Payments Pro

Website Payments Pro consists of APIs for accepting credit card, debit card, and PayPal payments; these payments can be immediate, authorized for later capture, and they can be recurring payments. Website Payments Pro also includes standalone applications for accepting payments.

In addition, Website Payments Pro includes Fraud Management Filters for automatic review and management of risk.

Settlements and Captured Payments

Often, you accept a payment and ship goods immediately, which is refered to as *sale*. In addition to immediate payments, Direct Payment and Express Checkout both allow you to authorize payments to be captured later, which is referred to as an *authorization*. An authorization is useful, for example, when you want to reserve a buyer's funds pending the shipment of goods; the actual payment is captured when the goods are shipped. An authorization can be reauthorized one time if necessary; for example, when you are unable to ship within 3 days of the authorization.

Express Checkout provides an additional option, called an *order*, which you use when a single authorization is insufficient. You can create multiple authorizations and capture them as part of the same order. This would be useful, for example, when an order is split into multiple shipments and you need to capture a payment each time part of the order is shipped.

Recurring Payments

You can support recurring payments to manage subscriptions and other payments on a fixed schedule. Direct Payment and Express Checkout both process recurring payments.

When you support recurring payments for a buyer, you create a *recurring payments profile*. The profile contains information about the recurring payments, including details for an optional trial period and a regular payment period. Both periods contain information about the payment frequency and payment amounts, including shipping and tax, if applicable.

After creating a profile, PayPal automatically queues payments based on the billing start date, billing frequency, and billing amount. Payments reoccur until the profile expires, there are too many failed payments to continue, or you cancel the profile.

Permission to allow recurring payments is established by the buyer setting up a billing agreement with the merchant on PayPal. For Express Checkout, the billing agreement can be established either in advance or when the buyer first makes a purchase; in either case, it occurs when you call Express Checkout API operations. For direct payment, it occurs when you make an explicit call to set up the billing agreement.

Recurring payments using reference transactions is an alternative, which enables you to handle payments for varying amounts of money on a varying schedule. A reference transaction is a financial transaction from which subsequent transactions can be derived; for example, a buyer can make a purchase on your site and the PayPal transaction ID, called a *reference transaction ID*, can later be used to initiate another transaction.

NOTE: The use of recurring payments with direct payment may incur additional fees.

Virtual Terminal

PayPal's Virtual Terminal is a web-based application that allows you to accept credit card payments. It is available to merchants in the United States, Canada, France, and the United Kingdom. Virtual Terminal provides your business with the functionality similar to a stand-alone credit card-processing terminal. Virtual Terminal is ideal when you receive orders by phone, fax, or by mail and want to accept credit cards. An optional card reader is available to process face-to-face purchases; however, some restrictions apply. You can use Virtual Terminal on any computer with an internet connection and a web browser.

For more information about Virtual Terminal, see Virtual Terminal Users Guide.

Hosted Solution

Hosted Solution, which is available to merchants in the United Kingdom as part of Website Payments Pro, is a fast and easy way to add transaction processing to your website. It is a secure, PayPal-hosted, web-based payment solution that allows you to securely send your buyers to PayPal's payment page to authorize and process transactions. Buyers pay with a debit or credit card, or their PayPal account. You do not have to capture or store credit card information on your website, thereby helping towards achieving PCI compliance. Hosted Solution is the choice for merchants who prefer a solution where all financial details are handled by PayPal.

For more information about Hosted Solution for UK merchants, see Website Payments Pro Hosted Solution Integration Guide.

Fraud Management Filters

Fraud Management Filters (FMF) provide you *filters* that identify potentially fraudulent transactions. There are 2 categories of filters:

- **Basic filters** screen against data such as the country of origin and the value of transactions. PayPal provides basic filters for Business accounts and Website Payments Pro accounts.
- **Advanced filters** screen data such as credit card and addresses information, lists of high-risk indicators, and additional transaction characteristics. Website Payments Pro merchants can upgrade to use these filters.

  NOTE: Using advanced filters might incur additional charges.

For more information about Fraud Management Filters, see Fraud Management Filters

Event Notification

In most cases you can use the `GetTransactionDetails` API operation to determine the information you need about a transaction. However, there may be some cases in which you must set up IPN; for example, when you need automatic notification about actions, such as disputes and their resolution.

IPN is a message service that PayPal uses to notify you about events, such as

- Instant payments, including Express Checkout, Adaptive Payments, and direct credit card payments, and authorizations, which indicate a sale whose payment has not yet been collected

- eCheck payments and associated status, such as *pending*, *completed*, or *denied*, and payments pending for other reasons, such as those being reviewed for potential fraud
- Recurring payment and subscription actions
- Chargebacks, disputes, reversals, and refunds associated with a transaction

For more information about IPN, see Instant Payment Notification Guide

Website Payments Pro API Operations

The PayPal API supports a range of functions related to payment processing. Though most API operations support both Direct Payment and Express Checkout, some are specific to Direct Payment and others are specific to Express Checkout.

PayPal API Operation	Description
Direct Payment core API operations: (Direct Payment only)	
`DoDirectPayment`	Process a credit card payment, such as a sale or authorization.
`DoNonReferencedCredit`	Issue a credit to a card not referenced by the original transaction. **NOTE:** Contact PayPal to use this API operation; in most cases, you should use the `RefundTransaction` API operation instead.
Express Checkout core API operations: (Express Checkout only)	
`SetExpressCheckout`	Initiates an Express Checkout transaction.
`GetExpressCheckoutDetails`	Obtain information about an Express Checkout transaction.
`DoExpressCheckoutPayment`	Completes an Express Checkout transaction.
Common API operations:	
`GetTransactionDetails`	Obtain information about a specific transaction.
`ManagePendingTransactionStatus`	Accept or deny a pending transaction held by Fraud Management Filters.
`RefundTransaction`	Issue a refund to the PayPal account holder associated with a transaction.
`TransactionSearch`	Search transaction history for transactions that meet the specified criteria.
Authorization and Capture API operations:	
`DoCapture`	Capture an authorized payment.
`DoAuthorization`	Authorize a payment. (Express Checkout only)

PayPal API Operation	Description
`DoReauthorization`	Reauthorize a previously authorized payment.
`DoVoid`	Void an order or an authorization.
Recurring Payment API operations:	
`CreateRecurringPaymentsProfile`	Create a recurring payments profile.
`GetRecurringPaymentsProfileDetails`	Obtain information about a recurring payments profile.
`ManageRecurringPaymentsProfileStatus`	Cancel, suspend, or reactivate a recurring payments profile.
`BillOutstandingAmount`	Bill the buyer for the outstanding balance associated with a recurring payments profile.
`UpdateRecurringPaymentsProfile`	Update a recurring payments profile.
`DoReferenceTransaction`	Process a payment from a buyer's account, which is identified by a previous transaction.
Recurring Payment Billing Agreement API operations: (Express Checkout only)	
`BAUpdate`	Update or delete a billing agreement.
`GetBillingAgreementCustomerDetails`	Obtain information about a billing agreement's PayPal account holder.
`SetCustomerBillingAgreement`	Initiates the creation of a billing agreement.
Other Express Checkout API operations: (Express Checkout only)	
`AddressVerify`	Confirms whether a postal address and postal code match those of the specified PayPal account holder. (Express Checkout only)
`Callback`	Define the shipping and handling parameters associated with Express Checkout.
`GetBalance`	Obtain the available balance for a PayPal account. (Express Checkout only)
`GetPalDetails`	Obtain your Pal ID, which is the PayPal-assigned merchant account number, and other information about your account.
`MassPay`	Make a payment to one or more PayPal account holders.

NOTE: If you use the Payflow API to process transactions with PayPal as your internet merchant account, see Website Payments Pro Payflow Edition Developers Guide.

Website Payments Pro Regional Differences

Website Payments Pro is available in the United States, Canada, and the United Kingdom. Minor regional differences include transaction limits, the kinds of credit cards accepted, and address information.

The following sections identify regional differences:

Credit Cards

The following table lists the credit cards that are accepted:

Country	Accepted credit cards
Canada	• Visa • MasterCard **NOTE:** Interac debit cards are not accepted.
United Kingdom	• Visa, including Visa Electron and Visa Debit • MasterCard • Maestro, including Switch • Solo
United States	• Visa • MasterCard • Discover • American Express

MasterCard is a registered trademark.

NOTE: For direct payment only, American Express restricts direct card acceptance merchants in certain business categories. Merchants are required to accept the American Express Card Acceptance agreement in order to process American Express cards directly.

Default Currency and Transaction Limits

The following table lists the default transaction limit in the default currency for each country:

Country	Default transaction limit in default currency
Canada	12,500 CAD
United Kingdom	5,500 GBP
United States	10,000 USD

NOTE: Contact PayPal if you want to increase transaction limits.

Credit Card Currencies by Country

The following currencies are supported for Direct Payment:

Currencies and Currency Codes Supported by Express Checkout and Direct Payment

Express Checkout Currency	Cur-rency Code	Direct Payment Currency for Specified Card in United States	Direct Payment Currency for Specified Card in United Kingdom	Direct Payment Currency for Specified Card in Canada
Australian Dollar	AUD	Visa, MasterCard	Visa, MasterCard	Visa, MasterCard
Canadian Dollar	CAD	Visa, MasterCard	Visa, MasterCard	Visa, MasterCard
Czech Koruna	CZK		Visa, MasterCard	Visa, MasterCard
Danish Krone	DKK		Visa, MasterCard	Visa, MasterCard
Euro	EUR	Visa, MasterCard	Visa, MasterCard	Visa, MasterCard
Hungarian Forint	HUF		Visa, MasterCard	Visa, MasterCard
Japanese Yen	JPY	Visa, MasterCard	Visa, MasterCard	Visa, MasterCard
Norwegian Krone	NOK		Visa, MasterCard	Visa, MasterCard
New Zealand Dollar	NZD		Visa, MasterCard	Visa, MasterCard
Polish Zloty	PLN		Visa, MasterCard	Visa, MasterCard
Pound Sterling	GBP	Visa, MasterCard	Visa, MasterCard, Maestro, Solo	Visa, MasterCard
Singapore Dollar	SGD		Visa, MasterCard	Visa, MasterCard
Swedish Krona	SEK		Visa, MasterCard	Visa, MasterCard
Swiss Franc	CHF		Visa, MasterCard	Visa, MasterCard
U.S. Dollar	USD	Visa, MasterCard, Discover, American Express	Visa, MasterCard	Visa, MasterCard

NOTE: Virtual Terminal for France supports the same currencies as Visa or MasterCard for the UK. Express Checkout supports all of the countries identified in the table.

Addresses

- For Canada, specify the province abbreviation in the State field.
- For Great Britain, the State field is ignored; however, you still may need to specify a value in the State field; for example you can specify the city for both the city and state.

For more information about addresses, see the PayPal Developer Network.

2 Introducing Direct Payment

Direct Payment lets buyers who do not have a PayPal account use their credit cards without leaving your website. PayPal processes the payment in the background.

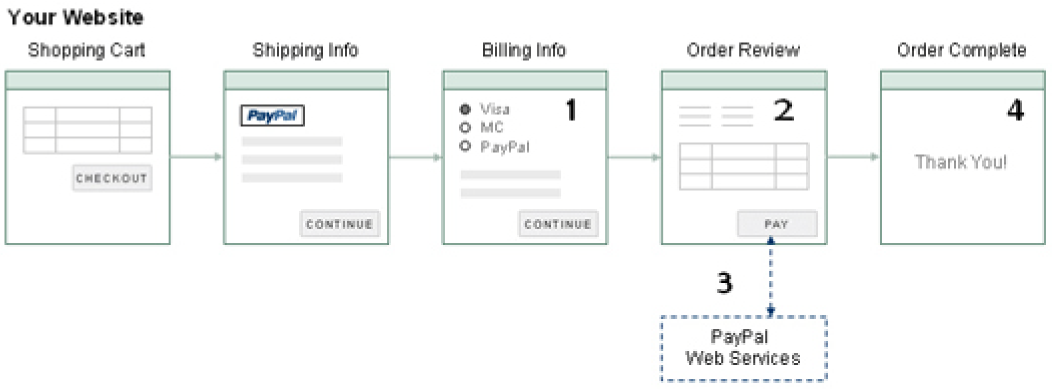
- The Direct Payment User Experience
- User Interface Recommendations for Direct Payment Checkout

The Direct Payment User Experience

Direct Payment enables buyers to pay by credit or debit card during your checkout flow. You have complete control over the experience; however, you must consider PCI compliance.

When buyers choose to pay with a credit or debit card, they enter their card number and other information on your website. After they confirm their order and click **Pay**, you complete the order in the background by invoking the `DoDirectPayment` API operation. Buyers never leave your site. Although PayPal processes the order, buyers aren't aware of PayPal's involvement; PayPal will not even appear on the buyer's credit card statement for the transaction.

The following diagram shows a typical Direct Payment flow:



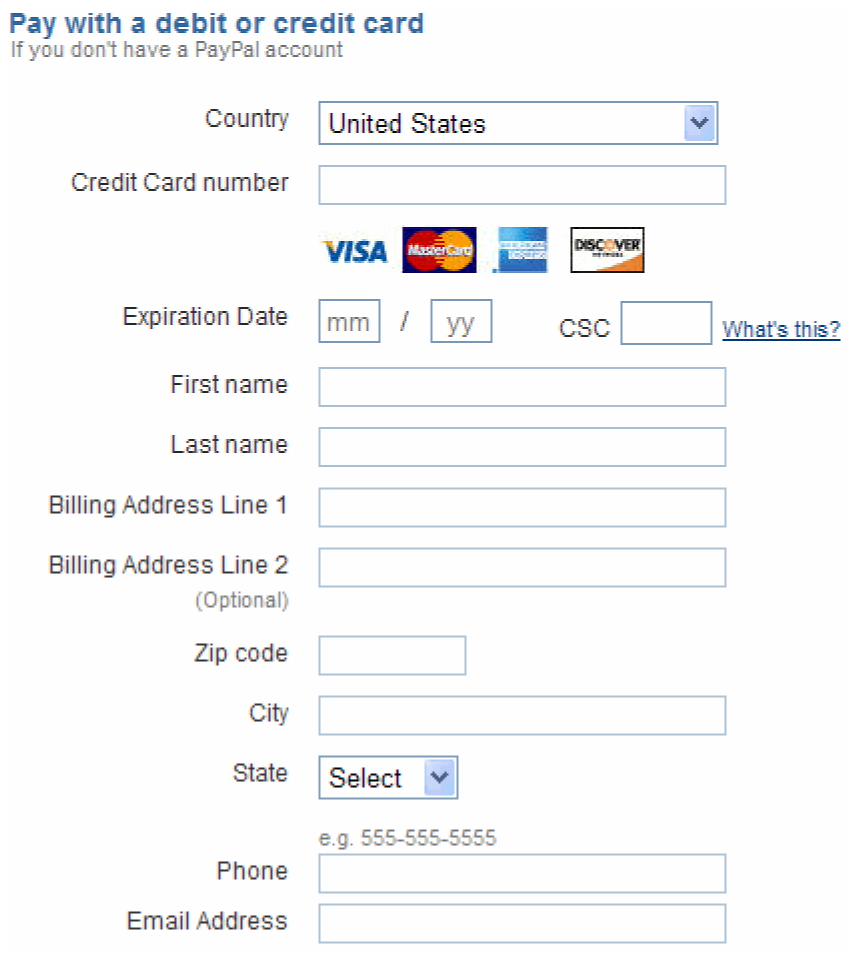
The numbers in the diagram correspond to the following implementation steps:

1. On your checkout pages, you need to collect the following information from a buyer to be used in the `DoDirectPayment` request:

   – Amount of the transaction
   – Credit card type
   – Credit card number
   – Credit card expiration date
   – Credit card CSC value

– Cardholder first and last name
– Cardholder billing address

The following example shows the collection of credit card information from a US buyer after the transaction amount has been determined:



Pay with a debit or credit card
If you don't have a PayPal account

Country	United States
Credit Card number	
	VISA MasterCard DISCOVER
Expiration Date	mm / yy CSC What's this?
First name	
Last name	
Billing Address Line 1	
Billing Address Line 2 (Optional)	
Zip code	
City	
State	Select
Phone	e.g. 555-555-5555
Email Address	

NOTE: In some cases, the billing address and CSC value may be optional. You must also identify *debit* on your PCI compliant checkout page when you reference a direct card checkout image.

2. You must also retrieve the IP address of the buyer's browser and include this with the request.

3. When a buyer clicks the **Pay** button, invoke the `DoDirectPayment` API operation.

4. The PayPal API server executes the request and returns a response.

– Ack code (Success, SuccessWithWarning, or Failure)
– Amount of the transaction
– AVS response code
– CSC response code

- PayPal transaction ID
- Error codes and messages (if any)
- Correlation ID (unique identifier for the API call)

5. If the operation is successful, you send the buyer to an order confirmation page.

The Ack code determines whether the operation is a success.

- If successful, you should display a message on the order confirmation page.
- Otherwise, you should show the buyer information related to the error. You should also provide an opportunity to pay using a different payment method.

User Interface Recommendations for Direct Payment Checkout

Your checkout pages must collect all the information you need to create the `DoDirectPayment` request. The request information can be collected by your site's checkout pages.

The following recommendations help process requests correctly and make it easier for buyers to provide necessary information:

IMPORTANT: You are responsible for processing card industry (PCI) compliance for protecting cardholder data. For example, storing the Card Security Code (CSC) violates PCI compliance. For more information about PCI compliance, see PCI Security Standards Council.

- Provide a drop-down menu for the *state* or *province* fields for addresses in countries that use them. For U.S. addresses, the state must be a valid 2-letter abbreviation for the state, military location, or U.S. territory. For Canada, the province must be a valid 2-letter province abbreviation. For the UK, do not use a drop-down menu; however, you may need to provide a value for the *state* in your `DoDirectPayment` request.
- Ensure buyers can enter the correct number of digits for the Card Security Code (CSC). The value is 3 digits for Visa, MasterCard, and Discover. The value is 4 digits for American Express.
- Show information on the checkout page that shows where to find the CSC code on the card and provide a brief explanation of its purpose.
- Configure timeout settings to allow for the fact that the `DoDirectPayment` API operation might take as long as 60 seconds to complete, even though completion in less than 3 seconds is typical. Consider displaying a "processing transaction" message to the buyer and disabling the **Pay** button until the transaction finishes.
- Use the optional Invoice ID field to prevent duplicate charges. PayPal ensures that an Invoice ID is used only once per account. Duplicate requests with the same Invoice ID result in an error and a failed transaction.

3 Introducing Express Checkout

Express Checkout is PayPal's premier checkout solution that streamlines the checkout process for buyers and keeps them on a merchant's website after making a purchase.

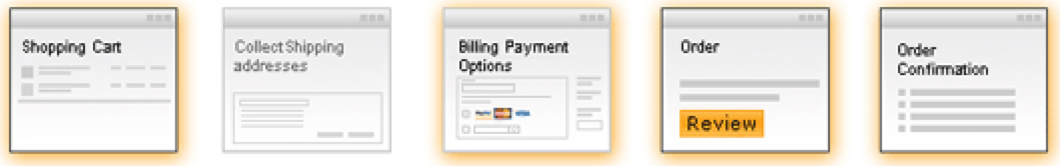
- The Express Checkout Experience
- Express Checkout Integration Steps
- Express Checkout Flow
- Express Checkout Building Blocks

The Express Checkout Experience

Express Checkout makes it easier for a buyers to pay online. It also enables you to accept PayPal while retaining control of the buyer and the overall checkout flow.

Consider your buyers' experience before implementing Express Checkout. A generic flow probably has the following sequence of pages:

A generic checkout flow



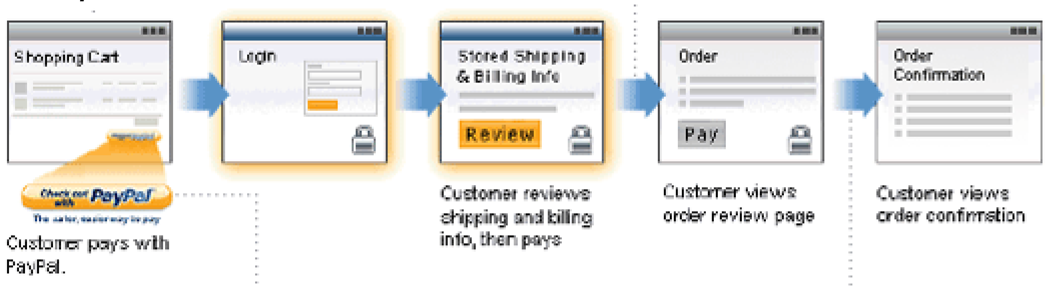
In a typical checkout flow, a buyer:

1. Checks out from the shopping cart page
2. Provides shipping information
3. Chooses a payment option and provides billing and payment information
4. Reviews the order and pays
5. Receives an order confirmation

In an Express Checkout flow, a buyer still checks out at the beginning of the flow. However, the buyer does not enter shipping, billing, or payment information, because PayPal provides the stored information. This simplifies and expedites the checkout process.

The following diagram shows the Express Checkout flow:

Express Checkout flow


Shopping Cart
Check out with PayPal
The safer, easier way to pay
Customer pays with PayPal.
Login
Stored Shipping & Billing Info
Review
Customer reviews shipping and billing info, then pays
Order
Pay
Customer views order review page
Order Confirmation
Customer views order confirmation

In the Express Checkout flow, the buyer:

1. Chooses Express Checkout by clicking **Check out with PayPal**
2. Logs into PayPal to authenticate his or her identity
3. Reviews the transaction on PayPal

   NOTE: Optionally, (not shown in the diagram), the buyer can then proceed to review the order on your site. You can also include other checkout steps, including upselling on your Review Order page.

4. Confirms the order and pays from your site
5. Receives an order confirmation

Express Checkout Integration Steps

You can implement Express Checkout in 4 steps:

1. Place PayPal checkout buttons and PayPal payment mark images in your checkout flow.
2. For each PayPal button that you place, modify your page to handle the button click.

   Use a PayPal Express Checkout API operation to set up the interaction with PayPal and redirect the browser to PayPal to initiate buyer approval for the payment.

3. On your Order page, use PayPal Express Checkout API operations to obtain the shipping address and accept the payment.
4. Test your integration using the PayPal Sandbox before taking your pages live.

Because PayPal offers you the flexibility to control your checkout flow, you should first understand how your current checkout flow works, then, become familiar with the Express Checkout flow. Start by reviewing Express Checkout Flow. For additional background information to help you get started, see Express Checkout Building Blocks.

Configuring and Customizing the Express Checkout Experience

After you implement and test your basic Express Checkout integration, you should configure the additional features of Express Checkout to customize it to meet your needs.

Carefully evaluate each feature because the more you streamline the checkout process and make Express Checkout seamless to buyers, the more likely your sales will increase.

At a minimum, you should:

- Set your logo on the PayPal site and provide order details in the transaction history.
- Use the PayPal confirmation page as your Order Review page to further streamline the user experience when you do not need the benefits associated with paying on your site. This strategy can lead to a better order completion rate, also known as a *conversion rate*.

Configure the look and feel of PayPal pages to match the look and feel of your site by specifying the:

- Logo to display
- Colors for the background and border
- Language in which PayPal content is displayed

You should include:

- Order details, including shipping and tax, during checkout

  IMPORTANT: Not displaying this information is a major cause of shopping cart abandonment during checkout.

- Shipping information for non-digital goods, which can be your address information for the buyer or the address on file with PayPal; if you use the address on file with PayPal, you can specify whether or not it must be a confirmed address

You can also activate additional features, including:

- Associate a payment with an eBay auction item
- Assign an invoice number to a payment
- Accept payments with giropay (Germany only)

Additional PayPal API Operations

You can use PayPal API operations to include advanced processing and back-office processes with Express Checkout. You can:

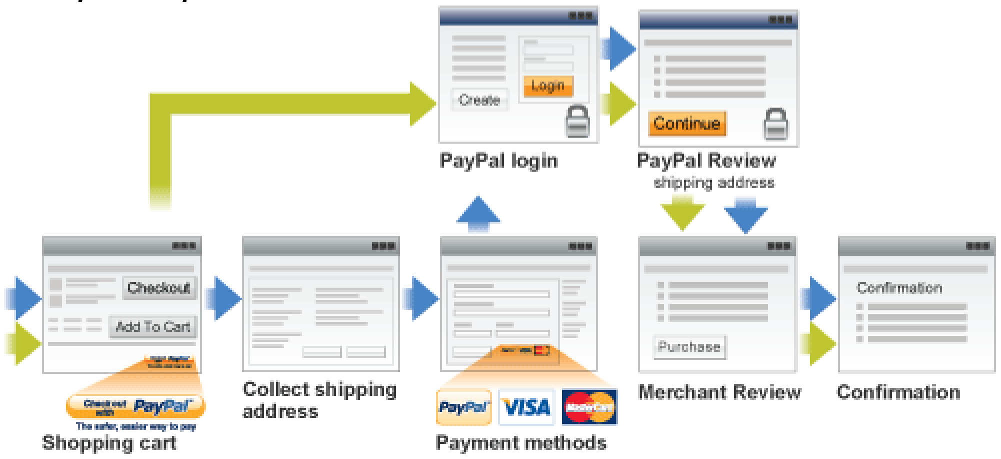
- Capture payments associated with authorizations and orders
- Process recurring payments
- Issue refunds, search transactions using various criteria, and provide other back-office operations

Express Checkout Flow

To implement Express Checkout, you must offer it both as a checkout option and as a payment method. Typically, you initiate the Express Checkout flow on your shopping cart page and on your payment options page.

You add Express Checkout to your existing flow by placing the **Checkout with PayPal** button on your **Shopping Cart** page and by placing the **PayPal** mark on your **Payment Methods** page. The following diagram shows the complete flow:

Complete Express Checkout flow



Make the following changes to implement the complete Express Checkout flow:

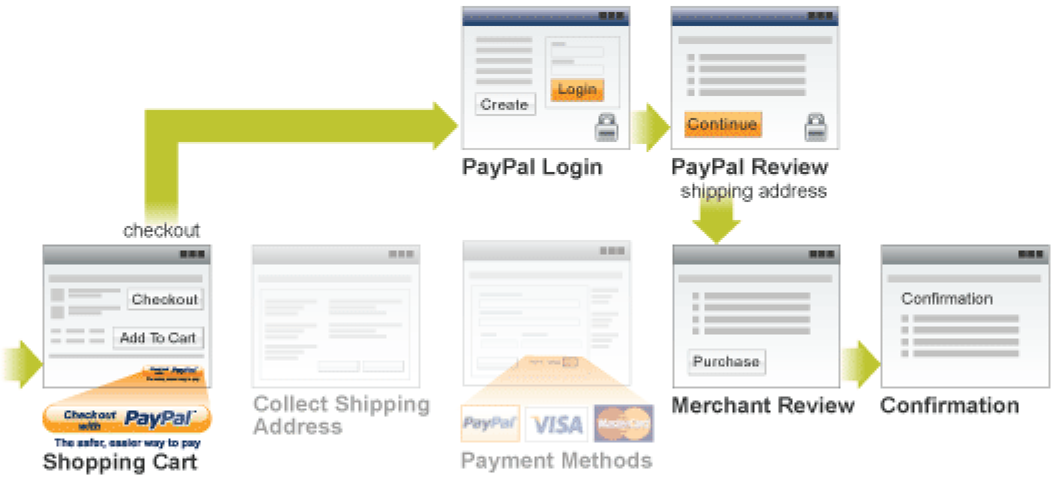
- On your **Shopping Cart** page, place the **Checkout with PayPal** button and respond to a click by setting up the Express Checkout request and redirecting your buyer's browser to PayPal.
- On your **Payment Methods** page, associate the **PayPal** mark with an option. Handle selection of the **PayPal** mark by setting up the Express Checkout request and redirecting your buyer's browser to PayPal.
- On the page your buyer returns to, obtain shipping information from PayPal and accept the payment to complete the Express Checkout transaction.

NOTE: You also can allow the buyer to pay on the PayPal Review page. In this case, your checkout flow can omit the Merchant Review page and proceed directly to your Confirmation page.

Checkout Entry Point

The checkout entry point is one of the places where you must implement Express Checkout. Buyers initiate the Express Checkout flow on your shopping cart page by clicking the **Checkout with PayPal** button.

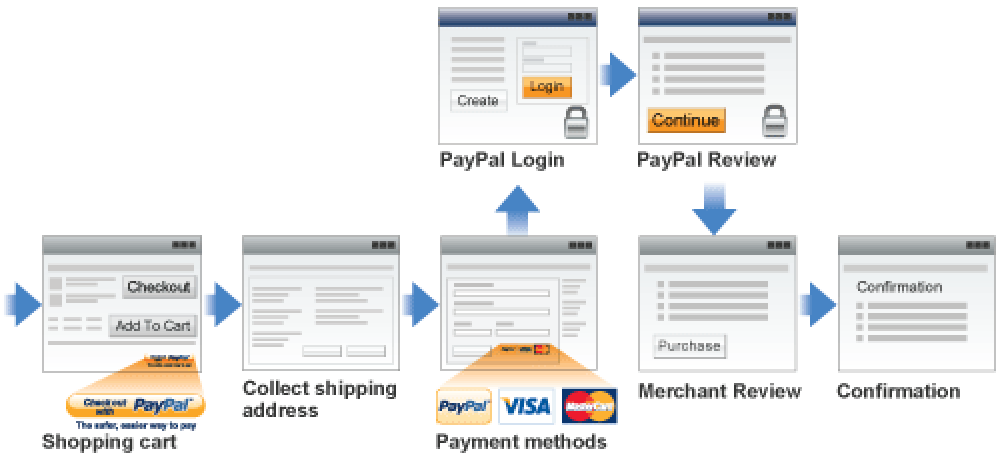
The following diagram shows how Express Checkout integrates with a typical checkout flow:



Payment Option Entry Point

The payment option entry point is one of the places where you must implement Express Checkout. Buyers initiate the Express Checkout flow on your payment methods page by selecting PayPal as the default option.

The following diagram shows how to integrate Express Checkout from your payment methods page:

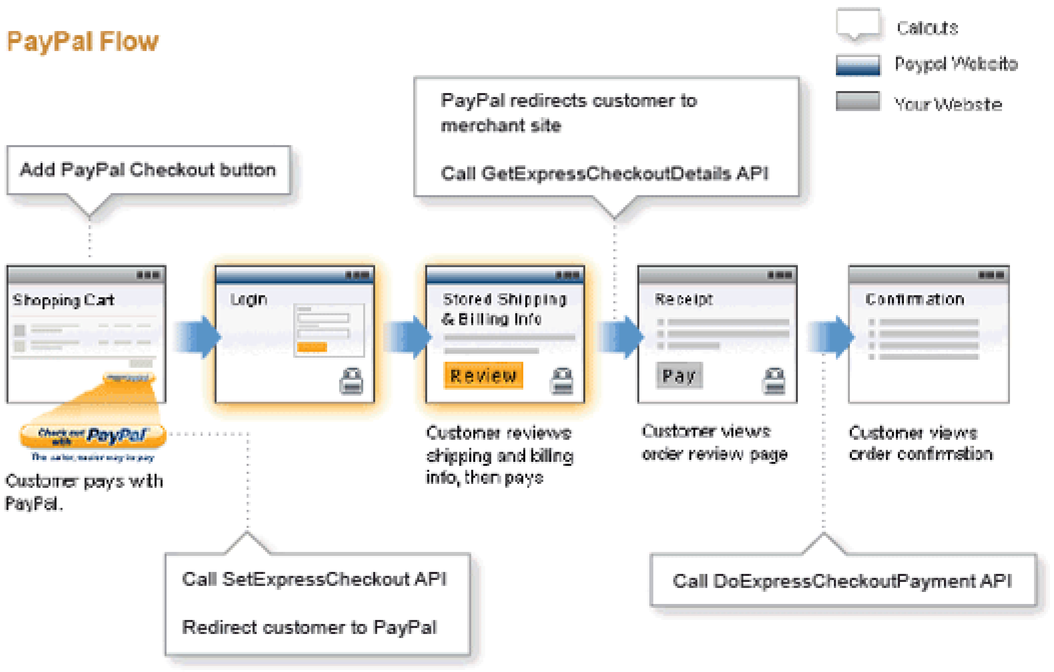


Express Checkout Building Blocks

You implement Express Checkout flows with Express Checkout buttons, PayPal API operations, PayPal commands, and tokens.

The following conceptual diagram identifies the building blocks that you use to integrate Express Checkout on your website:

Express Checkout Integration


PayPal Flow
Callouts
PayPal Website
Your Website
Add PayPal Checkout button
PayPal redirects customer to merchant site
Call GetExpressCheckoutDetails API
Shopping Cart
Login
Stored Shipping & Billing Info
Review
Receipt
Pay
Confirmation
Check out with PayPal
Customer pays with PayPal.
Customer reviews shipping and billing info, then pays
Customer views order review page
Customer views order confirmation
Call SetExpressCheckout API
Redirect customer to PayPal
Call DoExpressCheckoutPayment API

A *token* is a value assigned by PayPal that associates the execution of API operations and commands with a specific instance of a user experience flow.

NOTE: Tokens are not shown in the diagram.

PayPal Button and Logo Images

To inform buyers that PayPal is accepted on your website, you must place PayPal button and logo images in your checkout flow. PayPal recommends that you use dynamic images.

PayPal requires that you use the **Check out with PayPal** and the **PayPal** mark images hosted on secure PayPal servers. When the images are updated, the changes appear automatically in your application. Do not host copies of the PayPal images locally on your servers. Outdated PayPal images reduces buyer confidence in your site.

Express Checkout Image Flavors

The **Check out with PayPal** button and the **PayPal** mark images are available in two flavors:

- Dynamic image
- Static image

The dynamic images enable PayPal to change their appearance dynamically. If, for example, you have signed up to participate in a PayPal campaign, PayPal can change the appearance of the image dynamically for the duration of that campaign based on parameter information you append to the image URL. By default, the Express Checkout images appears as shown above.

The static images cannot be changed dynamically. To participate in a PayPal campaign, you would have to manually update the image code to change the image displayed and restore the default image when the campaign is over. The only way you can have image management taken care of for you is to replace static images in your implementation with dynamic images.

Express Checkout Images

The **Check out with PayPal** button is the image you place on your shopping cart page. The US version of the image looks like this.

To create an Express Checkout button, see https://www.paypal.com/us/cgi-bin/webscr?cmd=xpt/Merchant/merchant/ExpressCheckoutButtonCode-outside. PayPal also provides buttons for other countries. To locate a page for another country, replace the country abbreviation in the link with another country abbreviation. For example, replace `us` with `uk` for United Kingdom, as follows: https://www.paypal.com/uk/cgi-bin/webscr?cmd=xpt/Merchant/merchant/ExpressCheckoutButtonCode-outside. PayPal hosts images for the countries:

Country-specific buttons and images

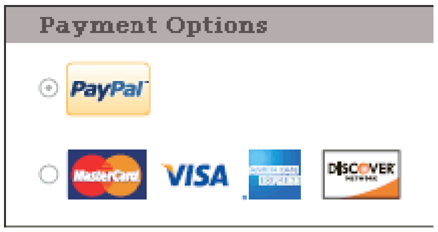
Country	URL Change	Country	URL Change	Country	URL Change	Country	URL Change
Australia	au	Austria	at	Belgium	be	Canada	ca
China	cn	France	fr	Germany	de	Italy	it
Japan	j1	Netherlands	nl	Poland	pl	Spain	es
Switzerland	ch	United Kingdom	uk	United States	us		

NOTE: URL changes are case sensitive. The abbreviation in the URL may not be a country code.

Payment Mark

The **PayPal** mark is the image you place on your payment methods page. It looks like this:

To implement PayPal as a payment option, which is part of the Express Checkout experience, associate the **PayPal** mark image with your payment options. PayPal recommends using radio buttons for payment options:



To create a **PayPal** mark, see https://www.paypal.com/cgi-bin/webscr?cmd=xpt/Marketing/general/OnlineLogoCenter-outside.

Express Checkout API Operations

The PayPal API provides three API operations for Express Checkout, which sets up the transaction, obtains information about the buyer, and handles the payment and completes the transaction.

API Operation	Description
`SetExpressCheckout`	Sets up the Express Checkout transaction. You can specify information to customize the look and feel of the PayPal site and the information it displays. You must include the following information: ● URL to the page on your website that PayPal redirects to after the buyer logs into PayPal and approves the payment successfully. ● URL to the page on your website that PayPal redirects to if the buyer cancels. ● Total amount of the order or your best estimate of the total. It should be as accurate as possible.
`GetExpressCheckout`	Obtains information about the buyer from PayPal, including shipping information.
`DoExpressCheckoutPayment`	Completes the Express Checkout transaction, including the actual total amount of the order.

Express Checkout Command

PayPal provides a command that you use when redirecting your buyer's browser to PayPal. This command enables your buyer to log into PayPal to approve an Express Checkout payment.

When you redirect your buyer's browser to PayPal, you must specify the `_ExpressCheckout` command for Express Checkout. You also specify the token that identifies the transaction, which was returned by the `SetExpressCheckout` API operation.

NOTE: To enable PayPal to redirect back to your website, you must have already invoked the `SetExpressCheckout` API operation, specifying URLs that PayPal uses to redirect back to your site. PayPal redirects to the *success* URL when the buyer pays on PayPal; otherwise, PayPal redirects to the *cancel* URL.

If the buyer approves the payment, PayPal redirects to the success URL with the following information:

- The token that was included in the redirect to PayPal
- The buyer's unique identifier (Payer ID)

If the buyer cancels, PayPal redirects to the cancel URL with the token that was included in the redirect to PayPal.

Express Checkout Token Usage

Express Checkout uses a token to control access to PayPal and execute Express Checkout API operations.

The `SetExpressCheckout` API operation returns a token, which is used by other Express Checkout API operations and by the `_ExpressCheckout` command to identify the transaction. The life of the token is approximately 3 hours.

4 Getting Started With Direct Payment

To get started with Direct Payment, implement and test the simplest `DoDirectPayment` API operation, which is a sale. Then you can expand your use of Direct Payment to include authorization and capture.

- Implementing the Simplest Direct Payment Integration
- Testing Direct Payment Using NVP and cURL
- Direct Payment Authorization and Captures

Implementing the Simplest Direct Payment Integration

To execute a direct payment transaction, you must invoke the `DoDirectPayment` API operation with sufficient information to identify the buyer's credit or debit card and the amount of the transaction.

This example assumes that you have set up the mechanism you will use to communicate with the PayPal server and have a PayPal business account with API credentials. It also assumes that the payment action is a final sale.

To set up the simplest direct payment transaction

1. Specify the amount of the transaction; include the currency if it is not in US dollars.

   Specify the total amount of the transaction if it is known; otherwise, specify the subtotal. Regardless of the specified currency, the format must have decimal point with exactly two digits to the right and an optional thousands separator to the left, which must be a comma; for example, EUR 2.000,00 must be specified as 2000.00 or 2,000.00. The specified amount cannot exceed USD $10,000.00, regardless of the currency used.

```
AMT=amount
CURRENCYCODE=currencyID
```

2. Specify the payment action.

   Although the default payment action is a `Sale`, it is a best practice to explicitly specify the payment action as one of the following values:

```
PAYMENTACTION=Sale
PAYMENTACTION=Authorization
```

3. Specify the IP address of the buyer's computer.

```
IPADDRESS=192.168.0.1
```

4. Specify information about the credit or debit card.

   You must specify the kind of credit or debit card and the account number:

   ```
   CREDITCARDTYPE=Visa
   ACCT=4683075410516684
   ```

   NOTE: The kind of card, the card issuer, and Payment Receiving Preferences settings in your PayPal profile may require you set additional fields:

   ```
   EXPDATE=042011
   CVV2=123
   ```

   NOTE: UK merchants must also specify values for 3D Secure-related fields when using Maestro.

5. Specify information about the card holder.

   You must specify the first and last name and the billing address associated with the card:

   ```
   FIRSTNAME=...
   LASTNAME=...
   STREET=...
   CITY=...
   STATE=...
   ZIP=...
   COUNTRYCODE=...
   ```

   NOTE: The state and zip code is not required for all countries.

Testing Direct Payment Using NVP and cURL

To test direct payment, you must first create a test business account in the Sandbox that is enabled for Website Payments Pro. You can then use the account to test credit and debit card payments using the `DoDirectPayment` API operation.

You can then simulate debit or credit card payments from cards that exist in the Sandbox. This example shows how to simulate a `DoDirectPayment` API operation using cURL to supply the NVP request values and to call `DoDirectPayment`.

To test Direct Payment in the Sandbox, you must first ensure that the Sandbox test account is associated with a credit card and enabled for Website Payments Pro.

The following example uses the `curl` command to execute the `DoDirectPayment` request and obtain a response. You can use the strategy shown in these steps for initial testing of your Direct Payment implementation. For more complete testing, you should integrate cURL into your checkout pages.

The following steps show how you can test the `DoDirectPayment` API operation:

1. Execute the `DoDirectPayment` API operation to complete the transaction.

The following example uses cURL to communicate with PayPal:

```
curl --insecure ^
https://api-3t.sandbox.paypal.com/nvp ^
-d "VERSION=56.0^
&SIGNATURE=api_signature^
&USER=api_username^
&PWD=api_password^
&METHOD=DoDirectPayment^
&PAYMENTACTION=Sale^
&IPADDRESS=192.168.0.1^
&AMT=8.88^
&CREDITCARDTYPE=Visa^
&ACCT=4683075410516684^
&EXPDATE=042011^
&CVV2=123^
&FIRSTNAME=John^
&LASTNAME=Smith^
&STREET=1 Main St.^
&CITY=San Jose^
&STATE=CA^
&ZIP=95131^
&COUNTRYCODE=US"
```

2. Test that the response to the `DoDirectPayment` API operation was successful.

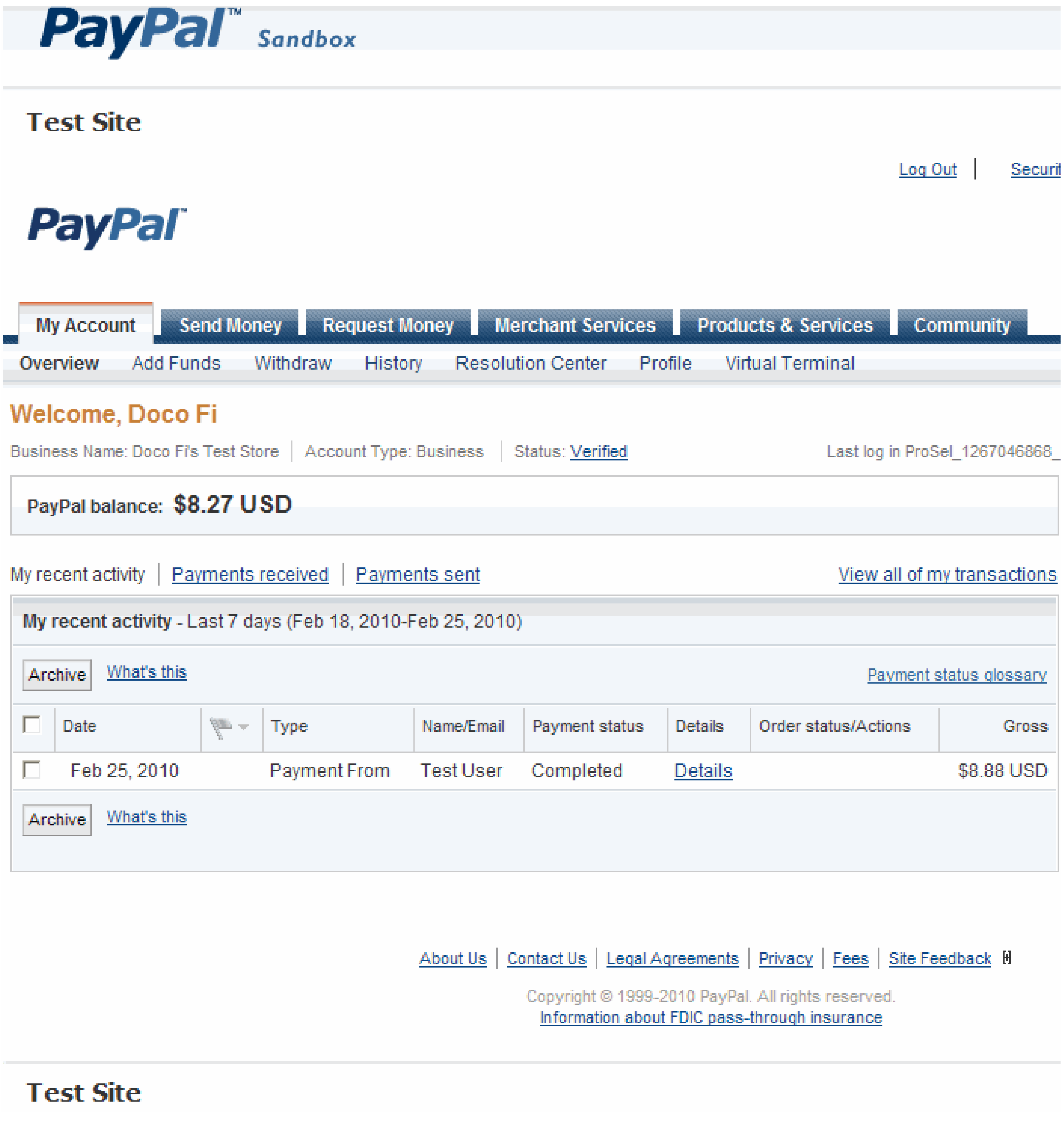
The ACK field must contain Success or `SuccessWithWarning`; however, other fields in the response can help you decide whether to ultimately accept or refund the payment:

```
TIMESTAMP=...
&CORRELATIONID=...
&ACK=Success
&VERSION=56%2e0
&BUILD=1195961
&AMT=8%2e88
&CURRENCYCODE=USD
&AVSCODE=X
&CVV2MATCH=M
&TRANSACTIONID=...
```

NOTE: Values of status fields are simulated because an actual card transaction does not occur.

3. Log into your PayPal test account from the Sandbox.

On the **My Account Overview** page, you should see the results of your transaction if the transaction was successful:



Welcome, Doco Fi

Business Name: Doco Fi's Test Store | Account Type: Business | Status: Verified   Last log in ProSel_1267046868_

PayPal balance: $8.27 USD

My recent activity | Payments received | Payments sent   View all of my transactions

My recent activity - Last 7 days (Feb 18, 2010-Feb 25, 2010)

Archive What's this   Payment status glossary

	Date		Type	Name/Email	Payment status	Details	Order status/Actions	Gross
☐	Feb 25, 2010		Payment From	Test User	Completed	Details		$8.88 USD

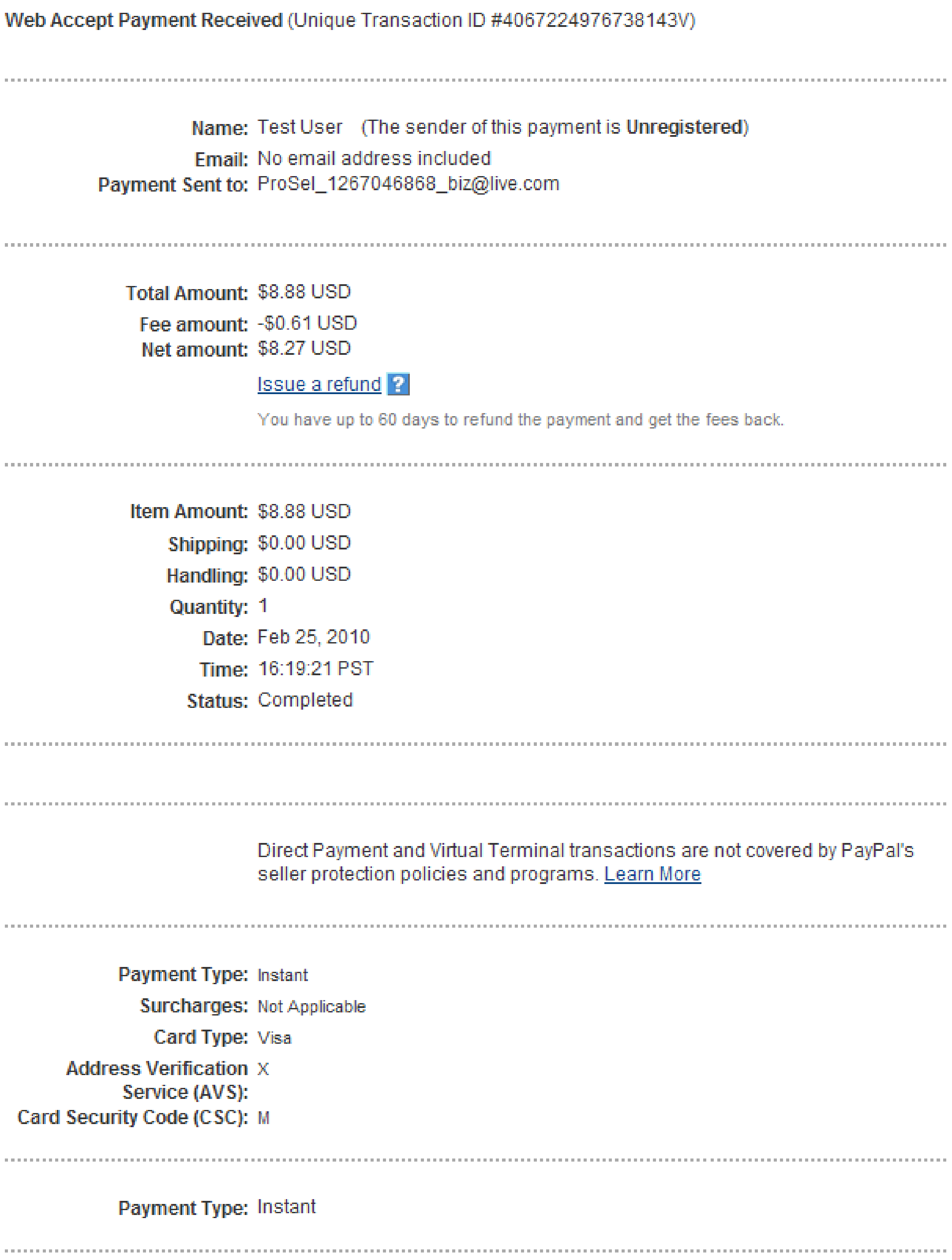
Archive What's this

About Us | Contact Us | Legal Agreements | Privacy | Fees | Site Feedback

Copyright © 1999-2010 PayPal. All rights reserved.
Information about FDIC pass-through insurance

Test Site

4. Click the Details link to see the status of the transaction.



Web Accept Payment Received (Unique Transaction ID #4067224976738143V)

Name:	Test User (The sender of this payment is **Unregistered**)
Email:	No email address included
Payment Sent to:	ProSel_1267046868_biz@live.com
Total Amount:	$8.88 USD
Fee amount:	-$0.61 USD
Net amount:	$8.27 USD
	Issue a refund ?
	You have up to 60 days to refund the payment and get the fees back.
Item Amount:	$8.88 USD
Shipping:	$0.00 USD
Handling:	$0.00 USD
Quantity:	1
Date:	Feb 25, 2010
Time:	16:19:21 PST
Status:	Completed

Direct Payment and Virtual Terminal transactions are not covered by PayPal's seller protection policies and programs. Learn More

Payment Type:	Instant
Surcharges:	Not Applicable
Card Type:	Visa
Address Verification Service (AVS):	X
Card Security Code (CSC):	M
Payment Type:	Instant

Direct Payment Authorization and Captures

You can use PayPal API operations to handle the capture of payments authorized using `DoDirectPayment`.

- Sale Payment Action for Direct Payment
- Authorization and Capture for Direct Payment
- DoDirectPayment Authorization and Capture Example
- DoDirectPayment Reauthorization and Capture Example

Sale Payment Action for Direct Payment

A *sale* payment action represents a single payment that completes a purchase for a specified amount.

A sale is the default payment action; however, you can also specify the action in your `DoDirectPayment` requests:

```
PAYMENTACTION=Sale
```

A sale is the most straightforward payment action. Choose this payment action if the transaction, including shipping of goods, can be completed immediately. Use this payment action when you intend to fulfill the order immediately, such as would be the case for digital goods or for items you have in stock for immediate shipment

After you execute the `DoDirectPayment` API operation, the payment is complete and further action is unnecessary. You cannot capture a further payment or void any part of the payment when you use this payment action.

Authorization and Capture for Direct Payment

An *authorization* payment action represents an agreement to pay and places the buyer's funds on hold for up to three days.

To set up an authorization, specify the following payment action in your `DoDirectPayment` request:

```
PAYMENTACTION=Authorization
```

An authorization enables you to capture multiple payments up to 115% of, or USD $75 more than, the amount you initially specify in the `DoDirectPayment` request. Choose this payment action if you need to ship the goods before capturing the payment or if there is some reason not to accept the payment immediately.

The *honor period*, for which funds can be held, is 3 days. The *valid period*, for which the authorization is valid, is 29 days. You can reauthorize the 3-day honor period at most once within the 29-day valid period.

Captures attempted outside of the honor period result in PayPal contacting the card issuer to reauthorize the payment; however, the reauthorization and, thus, the capture might be declined. If you know that you will capture after the honor period expires, PayPal recommends that you call `DoReauthorization` to explicitly reauthorize the honor period before attempting to capture the payment.

Consider an example in which the buyer orders three $100 items for a total of $300. You specify `Authorization` for your payment action in the `DoDirectPayment` request because the goods might not ship at the same time. For each of these items you call `DoCapture` to collect the payment.

On the first day, you ship the first item and on the third day you ship the second item. The payments for these items succeed because they occur within the honor period. If you ship the third item on the fifth day, your call to `DoCapture` might fail if the issuer declined the attempt by PayPal to reauthorize. Because the honor period has expired, you should explicitly call `DoReauthorization` before calling `DoCapture` to determine whether the buyer's funds are still available. Likewise, if all three items were to be shipped together after the honor period but before the valid period expires, you should explicitly call `DoReauthorization` before calling `DoCapture`.

When you call `DoCapture` for the final payment, you must set the `COMPLETETYPE` field to `Complete`. Prior calls to `DoCapture` must set this field to `NotComplete`. When payments are complete, any remaining uncaptured amount of the original authorization is automatically voided and nothing more can be captured.

You can explicitly void an authorization, in which case, the uncaptured part of the amount specified in the `DoDirectPayment` request becomes void and can no longer be captured. If no part of the payment has been captured, the entire payment becomes void and nothing can be captured. For Visa and MasterCard, a hold caused by the authorization is reversed; a bank hold might remain for 7 to 10 days until reversed by the card issuer.

NOTE: Authorizations become holds on a buyer's account that typically last 3 days for debit cards and 7 to 10 days for credit cards, depending on the issuer and region. If you decide not to capture an authorization, you should void the transaction, which triggers an authorization reversal.

API operations associated with the Authorization payment action in DoDirectPayment

API Operation	Description
`DoCapture`	Capture an authorized payment
`DoReauthorization`	Reauthorize a payment
`DoVoid`	Void an authorization

DoDirectPayment Authorization and Capture Example

This example authorizes a credit card payment using the `DoDirectPayment` API operation and then uses the `DoCapture` API operation to capture the payment.

IMPORTANT: Never use post actions for live transactions; they are not secure.

In your `DoDirectPayment` request, set the payment action to `Authorization`:

```
<form method=post action=https://sandbox.paypal.com/nvp>
<input type=hidden name=USER value=...>
<input type=hidden name=PWD value=...>
<input type=hidden name=SIGNATURE value=...>
<input type=hidden name=VERSION value= 58.0>
<input type=hidden name=PAYMENTACTION value=Authorization>
<input type=hidden name=CREDITCARDTYPE value=Visa>
<input type=hidden name=ACCT value=...>
<input type=hidden name=STARTDATE value=112000>
<input type=hidden name=EXPDATE value=112020>
<input type=hidden name=CVV2 value=123>
<input type=hidden name=AMT value=20.00>
<input type=hidden name=CURRENCYCODE value=USD>
<input type=hidden name=FIRSTNAME value=...>
<input type=hidden name=LASTNAME value=...>
<input type=hidden name=STREET value=...>
<input type=hidden name=STREET2  value=>
<input type=hidden name=CITY value="San Francisco">
<input type=hidden name=STATE value=CA>
<input type=hidden name=Zip value=94121>
<input type=hidden name=COUNTRYCODE value=US>
<input type=hidden name=EMAIL value=... >
<input type=submit name=METHOD value=DoDirectPayment>
</form>
```

If the authorization was successful, the response contains the authorization ID in the transaction ID field:

```
TIMESTAMP=2010%2d03%2d08T19%3a35%3a18Z&CORRELATIONID=ab12f37f9566&ACK=Succe
ss&VERSION=58%2e0&BUILD=1218643&AMT=20%2e00&CURRENCYCODE=USD&AVSCODE=X&CVV2
MATCH=M&TRANSACTIONID=6RH38738S17889722
```

Use this ID in the `DoCapture` request to specify the authorization that you want to capture:

```
<form method=post action=https://sandbox.paypal.com/nvp>
<input type=hidden name=USER value=...>
<input type=hidden name=PWD value=...>
<input type=hidden name=SIGNATURE value=...>
<input type=hidden name=VERSION value= 58.0>
<input type=hidden name=AUTHORIZATIONID value=6RH38738S17889722>
<input type=hidden name=AMT value=5>
<input type=hidden name=CURRENCYCODE value=USD>
<input type=hidden name=COMPLETETYPE value=Complete>
<input type=hidden name=INVNUM value=>
<input type=hidden name=NOTE value= March 08 2010>
<input type=hidden name=SOFTDESCRIPTOR value=>
<input type=submit name=METHOD value=DoCapture>
</form>
```

If the capture was successful, the payment status is `Completed`:

```
AUTHORIZATIONID=6RH38738S17889722&TIMESTAMP=2010%2d03%2d08T19%3a47%3a39Z&CO
RRELATIONID=d1e8043ae0a12&ACK=Success&VERSION=58%2e0&BUILD=1218643&TRANSACT
IONID=5F62121256435650V&PARENTTRANSACTIONID=6RH38738S17889722&RECEIPTID=007
8%2d2642%2d6061%2d5728&TRANSACTIONTYPE=webaccept&PAYMENTTYPE=instant&EXPECT
EDECHECKCLEARDATE=1970%2d01%2d01T00%3a00%3a00Z&ORDERTIME=2010%2d03%2d08T19%
3a47%3a38Z&AMT=5%2e00&FEEAMT=0%2e45&TAXAMT=0%2e00&CURRENCYCODE=USD&PAYMENTS
TATUS=Completed&PENDINGREASON=None&REASONCODE=None&PROTECTIONELIGIBILITY=In
eligible
```

NOTE: The `DoCapture` response returns a new transaction ID as well as the authorization ID. The authorization ID is also the parent transaction ID for the completed transaction.

DoDirectPayment Reauthorization and Capture Example

This example authorizes a credit card payment using the `DoDirectPayment` API operation and then reauthorizes the payment using the `DoReauthorization` API operation before capturing it with the `DoCapture` API operation.

IMPORTANT: Never use post actions for live transactions; they are not secure.

In your `DoDirectPayment` request, set the payment action to `Authorization`:

```
<form method=post action=https://sandbox.paypal.com/nvp>
<input type=hidden name=USER value=...>
<input type=hidden name=PWD value=...>
<input type=hidden name=SIGNATURE value=...>
<input type=hidden name=VERSION value= 58.0>
<input type=hidden name=PAYMENTACTION value=Authorization>
<input type=hidden name=CREDITCARDTYPE value=Visa>
<input type=hidden name=ACCT value=...>
<input type=hidden name=STARTDATE value=112000>
<input type=hidden name=EXPDATE value=112020>
<input type=hidden name=CVV2 value=123>
<input type=hidden name=AMT value=500>
<input type=hidden name=CURRENCYCODE value=USD>
<input type=hidden name=FIRSTNAME value=...>
<input type=hidden name=LASTNAME value=...>
<input type=hidden name=STREET value=...>
<input type=hidden name=STREET2 value=>
<input type=hidden name=CITY value="San Francisco">
<input type=hidden name=STATE value=CA>
<input type=hidden name=Zip value=94121>
<input type=hidden name=COUNTRYCODE value=US>
<input type=hidden name=EMAIL value=... >
<input type=submit name=METHOD value=DoDirectPayment>
</form>
```

If the authorization was successful, the response contains the authorization ID in the transaction ID field:

```
TIMESTAMP=2010%2d03%2d05T03%3a55%3a13Z&CORRELATIONID=2f8b1e854983e&ACK=Succ
ess&VERSION=62%2e0&BUILD=1218643&AMT=500%2e00&CURRENCYCODE=USD&AVSCODE=X&CV
V2MATCH=M&TRANSACTIONID=4HS1916972552122T
```

Use the transaction ID in the `DoReauthorization` request to identify the authorization that you want to reauthorize:

```
<form method=post action=https://sandbox.paypal.com/nvp>
<input type=hidden name=USER value=...>
<input type=hidden name=PWD value=...>
<input type=hidden name=SIGNATURE value=...>
<input type=hidden name=VERSION value= 62.0>
<input type=hidden name=AUTHORIZATIONID value=4HS1916972552122T>
<input type=hidden name=AMT value=23>
<input type=hidden name=CURRENCYCODE value=USD>
<input type=submit name=METHOD value=DoReauthorization>
</form>
```

The response to `DoReauthorization` contains a new authorization ID:

```
AUTHORIZATIONID=6HB59926VL998415S&TIMESTAMP=2010%2d03%2d08T20%3a37%3a48Z&CO
RRELATIONID=797da6e380c0&ACK=Success&VERSION=62%2e0&BUILD=1218643&PAYMENTST
ATUS=Pending&PENDINGREASON=authorization&PROTECTIONELIGIBILITY=Ineligible
```

Use the new authorization ID in the `DoCapture` request:

```
<form method=post action=https://sandbox.paypal.com/nvp>
<input type=hidden name=USER value=...>
<input type=hidden name=PWD value=...>
<input type=hidden name=SIGNATURE value=...>
<input type=hidden name=VERSION value= 62.0>
<input name=AUTHORIZATIONID value=6HB59926VL998415S>
<input name=AMT value=45>
<input name=CURRENCYCODE value=USD>
<input name=COMPLETETYPE value=Complete>
<input name=INVNUM value=>
<input name=NOTE value=>
<input name=SOFTDESCRIPTOR value=>
<input type=submit name=METHOD value=DoCapture>
</form>
```

If the capture was successful, the payment status is `Completed`:

```
AUTHORIZATIONID=6HB59926VL998415S&TIMESTAMP=2010%2d03%2d08T21%3a06%3a01Z&CO
RRELATIONID=8955b8704da91&ACK=Success&VERSION=62%2e0&BUILD=1218643&TRANSACT
IONID=2BG77878LE143642C&PARENTTRANSACTIONID=4HS1916972552122T&RECEIPTID=111
5%2d8794%2d3120%2d6892&TRANSACTIONTYPE=webaccept&PAYMENTTYPE=instant&EXPECT
EDECHECKCLEARDATE=1970%2d01%2d01T00%3a00%3a00Z&ORDERTIME=2010%2d03%2d08T21%
3a06%3a00Z&AMT=45%2e00&FEEAMT=1%2e61&TAXAMT=0%2e00&CURRENCYCODE=USD&PAYMENT
STATUS=Completed&PENDINGREASON=None&REASONCODE=None&PROTECTIONELIGIBILITY=I
neligible
```

NOTE: The `DoCapture` response returns a new transaction ID as well as the authorization ID associated with the reauthorization. The authorization ID is also the parent transaction ID for the completed transaction.

5 Getting Started With Express Checkout

To implement Express Checkout, start with the simplest Express Checkout integration and test it. Then you can decide the kind of payment settlement actions you want to support.

- Implementing the Simplest Express Checkout Integration
- Testing an Express Checkout Integration
- Handling Payment Settlements With Express Checkout
- Issuing Refunds

Implementing the Simplest Express Checkout Integration

The simplest Express Checkout integration requires the following PayPal API operations: `SetExpressCheckout`, `DoExpressCheckoutPayment`, and optionally, `GetExpressCheckoutDetails`.

- Setting Up the Express Checkout Transaction
- Obtaining Express Checkout Transaction Details
- Completing the Express Checkout Transaction

Setting Up the Express Checkout Transaction

To set up an Express Checkout transaction, you must invoke the `SetExpressCheckout` API operation to provide sufficient information to initiate the payment flow and redirect to PayPal if the operation was successful.

This example assumes that you have set up the mechanism you will use to communicate with the PayPal server and have a PayPal business account with API credentials. It also assumes that the payment action is a final sale.

When you set up an Express Checkout transaction, you specify values in the `SetExpressCheckout` request and then call the API. The values you specify control the PayPal page flow and the options available to you and your buyers. You should start by setting up a standard Express Checkout transaction, which can be modified to include additional options.

To set up the simplest standard Express Checkout transaction

1. Specify the amount of the transaction; include the currency if it is not in US dollars.

   Specify the total amount of the transaction if it is known; otherwise, specify the subtotal. Regardless of the specified currency, the format must have decimal point with exactly two digits to the right and an optional thousands separator to the left, which must be a comma.

For example, EUR 2.000,00 must be specified as 2000.00 or 2,000.00. The specified amount cannot exceed USD $10,000.00, regardless of the currency used.

```
PAYMENTREQUEST_0_AMT=amount
PAYMENTREQUEST_0_CURRENCYCODE=currencyID
```

2. Specify the return URL.

   The return URL is the page to which PayPal redirects your buyer's browser after the buyer logs into PayPal and approves the payment. Typically, this is a secure page (`https://...`) on your site.

   NOTE: You can use the return URL to piggyback parameters between pages on your site. For example, you can set your Return URL to specify additional parameters using the `https://www.`*yourcompany*`.com/`*page*`.html?`*param=value...* syntax. The parameters become available as request parameters on the page specified by the Return URL.

```
RETURNURL=return_url
```

3. Specify the cancel URL.

   The cancel URL is the page to which PayPal redirects your buyer's browser if the buyer does not approve the payment. Typically, this is the secure page (`https://...`) on your site from which you redirected the buyer to PayPal.

   NOTE: You can pass `SetExpressCheckout` request values as parameters in your URL to have the values available, if necessary, after PayPal redirects to your URL.

```
CANCELURL=cancel_url
```

4. Specify the payment action.

   Although the default payment action is a `Sale`, it is a best practice to explicitly specify the payment action as one of the following values:

```
PAYMENTREQUEST_0_PAYMENTACTION=Sale
PAYMENTREQUEST_0_PAYMENTACTION=Authorization
PAYMENTREQUEST_0_PAYMENTACTION=Order
```

5. Execute the `SetExpressCheckout` API operation to set up the Express Checkout transaction.

6. Test that the response to the `SetExpressCheckout` API operation was successful.
7. If calling the `SetExpressCheckout` API was successful, redirect the buyer's browser to PayPal and execute the `_express-checkout` command using the token returned in the `SetExpressCheckout` response.

NOTE: The following example uses the PayPal Sandbox server:

```
https://www.sandbox.paypal.com/webscr
        ?cmd=_express-checkout&token=tokenValue
```

Obtaining Express Checkout Transaction Details

To obtain details about an Express Checkout transaction, you can invoke the `GetExpressCheckoutDetails` API operation.

This example assumes that PayPal redirects to your buyer's browser with a valid token after the buyer reviews the transaction on PayPal.

Although you are not required to invoke the `GetExpressCheckoutDetails` API operation, most Express Checkout implementations take this action to obtain information about the buyer. You invoke the `GetExpressCheckoutDetails` API operation from the page specified by return URL, which you set in your call to the `SetExpressCheckout` API. Typically, you invoke this operation as soon as the redirect occurs and use the information in the response to populate your review page.

To obtain a buyer's shipping address and Payer ID

1. Specify the token returned by PayPal when it redirects the buyer's browser to your site.

   PayPal returns the token to use in the `token` HTTP request parameter when redirecting to the URL you specified in your call to the `SetExpressCheckout` API.

   ```
   TOKEN=tokenValue
   ```

2. Execute the `GetExpressCheckoutDetails` API to obtain information about the buyer.
3. Access the fields in the `GetExpressCheckoutDetails` API response.

   NOTE: Only populated fields are returned in the response.

Completing the Express Checkout Transaction

To complete an Express Checkout transaction, you must invoke the `DoExpressCheckoutPayment` API operation.

This example assumes that PayPal redirects your buyer's browser to your website with a valid token after you call the `SetExpressCheckout` API. Optionally, you may call the

`GetExpressCheckoutDetails` API before calling the `DoExpressCheckoutPayment` API.

In the simplest case, you set the total amount of the order when you call the `SetExpressCheckout` API. However, you can change the amount before calling the `DoExpressCheckoutPayment` API if you did not know the total amount when you called the `SetExpressCheckout` API.

This example assumes the simplest case, in which the total amount was specified in the return URL when calling the `SetExpressCheckout` API. Although you can specify additional options, this example does not use any additional options.

To execute an Express Checkout transaction

1. Specify the token returned by PayPal when it redirects the buyer's browser to your site.

   PayPal returns the token to use in the `token` HTTP request parameter when redirecting to the URL you specified in your call to the `SetExpressCheckout` API.

   ```
   TOKEN=tokenValue
   ```

2. Specify the Payer ID returned by PayPal when it redirects the buyer's browser to your site.

   PayPal returns the Payer ID to use in the `token` HTTP request parameter when redirecting to the URL you specified in your call to the `SetExpressCheckout` API. Optionally, you can obtain the Payer ID by calling the `GetExpressCheckoutDetails` API.

   ```
   PAYERID=id
   ```

3. Specify the amount of the order including shipping, handling, and tax; include the currency if it is not in US dollars.

   Regardless of the specified currency, the format must have decimal point with exactly two digits to the right and an optional thousands separator to the left, which must be a comma; for example, EUR 2.000,00 must be specified as 2000.00 or 2,000.00. The specified amount cannot exceed USD $10,000.00, regardless of the currency used.

   ```
   PAYMENTREQUEST_0_AMT=amount
   PAYMENTREQUEST_0_CURRENCYCODE=currencyID
   ```

4. Specify the payment action.

   Although the default payment action is a `Sale`, it is a best practice to explicitly specify the payment action as one of the following values:

   ```
   PAYMENTREQUEST_0_PAYMENTACTION=Sale
   PAYMENTREQUEST_0_PAYMENTACTION=Authorization
   PAYMENTREQUEST_0_PAYMENTACTION=Order
   ```

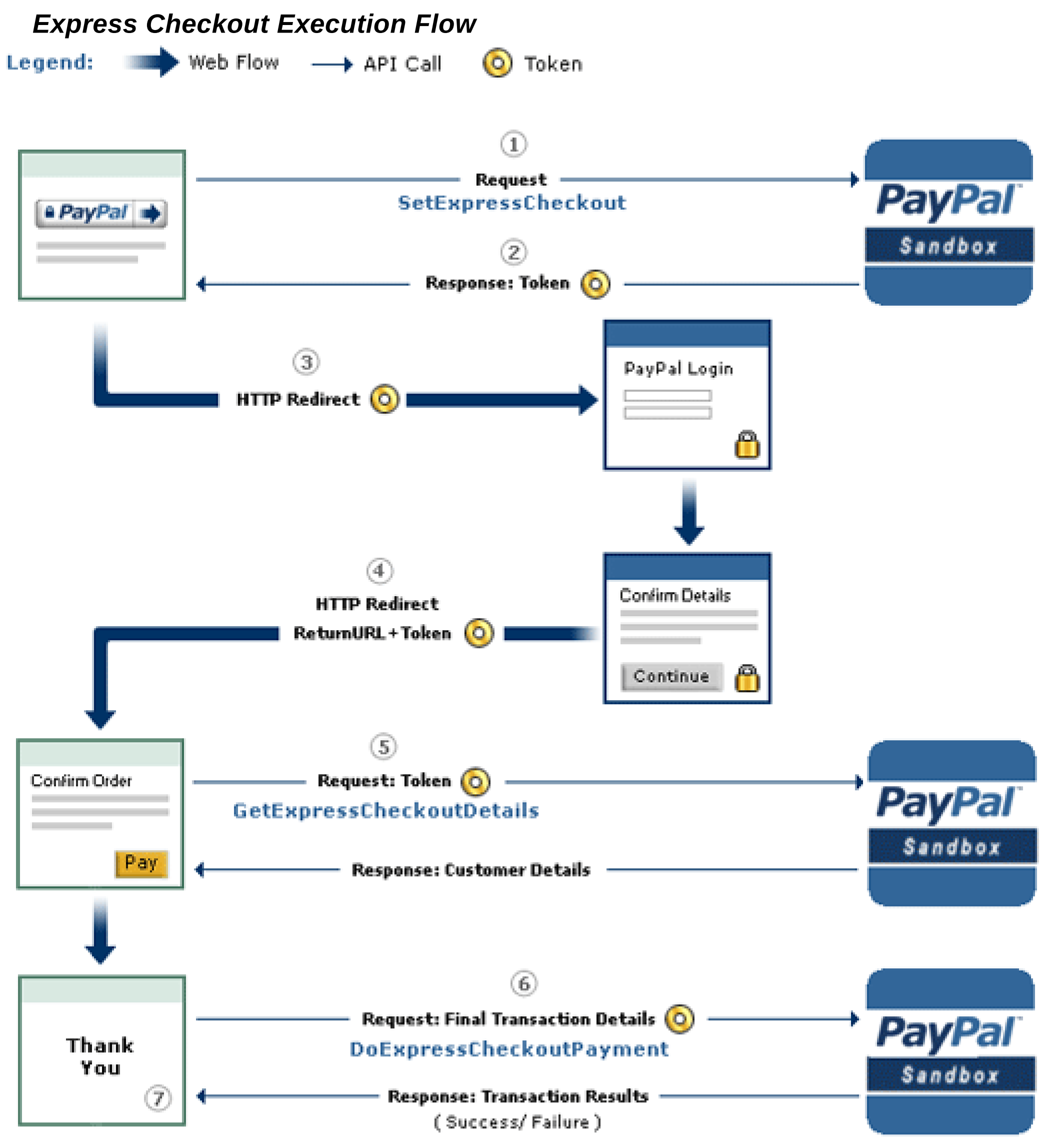
5. Execute the `DoExpressCheckoutPayment` API to complete the Express Checkout transaction.
6. Examine the values returned by the API if the transaction completed successfully.

Testing an Express Checkout Integration

You can test your Express Checkout integration in the Sandbox.

This example shows how to simulate your web pages using HTTP forms and supplying the values for API operations from these forms. You can use this strategy for your initial testing; however, for more complete testing, you will want to replace these forms with your web pages containing actual code.

The following diagram shows the Express Checkout execution flow, which uses the Sandbox as the API server. The pages on the left represent your site.


Express Checkout Execution Flow
Legend: Web Flow API Call Token
PayPal
1 Request SetExpressCheckout
2 Response: Token
PayPal Sandbox
3 HTTP Redirect
PayPal Login
4 HTTP Redirect ReturnURL + Token
Confirm Details
Continue
Confirm Order
Pay
5 Request: Token GetExpressCheckoutDetails
Response: Customer Details
6 Request: Final Transaction Details DoExpressCheckoutPayment
Response: Transaction Results (Success/ Failure)
Thank You
7

The following steps match the circled numbers in the diagram. Perform the actions in each step to test Express Checkout.

1. Invoke a form on your site that calls the `SetExpressCheckout` API on the Sandbox.

To invoke the API, set form fields whose names match the NVP names of the fields you want to set, specify their corresponding values, and then post the form to a PayPal Sandbox server, such as `https://api-3t.sandbox.paypal.com/nvp`, as shown in the following example:

```
<form method=post action=https://api-3t.sandbox.paypal.com/nvp>
   <input type=hidden name=USER value=API_username>
   <input type=hidden name=PWD value=API_password>
   <input type=hidden name=SIGNATURE value=API_signature>
   <input type=hidden name=VERSION value=XX.0>
   <input type=hidden name=PAYMENTREQUEST_0_PAYMENTACTION
      value=Authorization>
   <input name=PAYMENTREQUEST_0_AMT value=19.95>
   <input type=hidden name=RETURNURL
      value=http://www.YourReturnURL.com>
   <input type=hidden name=CANCELURL
      value=http://www.YourCancelURL.com>
   <input type=submit name=METHOD value=SetExpressCheckout>
</form>
```

NOTE: The API username is a Sandbox business test account for which a signature exists. See the Test Certificates tab of the Sandbox to obtain a signature. If you are not using a signature, you must use a different Sandbox server.

2. Review the response string from the `SetExpressCheckout` API operation.

   PayPal responds with a message, such as the one shown below. Note the status, which should include `ACK` set to `Success`, and a token that is used in subsequent steps.

```
TIMESTAMP=2007%2d04%2d05T23%3a23%3a07Z
&CORRELATIONID=63cdac0b67b50
&ACK=Success
&VERSION=XX%2e000000
&BUILD=1%2e0006
&TOKEN=EC%2d1NK66318YB717835M
```

3. If the operation was successful, use the token and redirect your browser to the Sandbox to log in, as follows:

```
https://www.sandbox.paypal.com/cgi-bin/webscr?
cmd=_express-checkout
&token=EC-1NK66318YB717835M
```

   You may need to replace hexadecimal codes with ASCII codes; for example, you may need to replace `%2d` in the token with a hyphen (`-`).

   You must log in to `https://developer.paypal.com` before you log in to a Sandbox test account. You then log in to the test account that represents the buyer, not the API_username business test account that represents you as the merchant.

4. After logging into the buyer test account, confirm the details.

   When you confirm, the Sandbox redirects your browser to the return URL you specified when invoking the `SetExpressCheckout` API operation, as in the following example:

```
http://www.YourReturnURL.com/
                    ?token=EC-1NK66318YB717835M&PayerID=7AKUSARZ7SAT8
```

5. Invoke a form on your site that calls the `GetExpressCheckoutDetails` API operation on the Sandbox:

```
<form method=post action=https://api-3t.sandbox.paypal.com/nvp
   <input type=hidden name=USER value=API_username>
   <input type=hidden name=PWD value=API_password>
   <input type=hidden name=SIGNATURE value=API_signature>
   <input type=hidden name=VERSION value=XX.0>
   <input name=TOKEN value=EC-1NK66318YB717835M>
   <input type=submit name=METHOD value=GetExpressCheckoutDetails>
</form>
```

If the operation was successful, the `GetExpressCheckoutDetails` API returns information about the payer, such as the following information:

```
TIMESTAMP=2007%2d04%2d05T23%3a44%3a11Z
&CORRELATIONID=6b174e9bac3b3
&ACK=Success
&VERSION=XX%2e000000
&BUILD=1%2e0006
&TOKEN=EC%2d1NK66318YB717835M
&EMAIL=YourSandboxBuyerAccountEmail
&PAYERID=7AKUSARZ7SAT8
&PAYERSTATUS=verified
&FIRSTNAME=...
&LASTNAME=...
&COUNTRYCODE=US
&BUSINESS=...
&PAYMENTREQUEST_0_SHIPTONAME=...
&PAYMENTREQUEST_0_SHIPTOSTREET=...
&PAYMENTREQUEST_0_SHIPTOCITY=...
&PAYMENTREQUEST_0_SHIPTOSTATE=CA
&PAYMENTREQUEST_0_SHIPTOCOUNTRYCODE=US
&PAYMENTREQUEST_0_SHIPTOCOUNTRYNAME=United%20States
&PAYMENTREQUEST_0_SHIPTOZIP=94666
&PAYMENTREQUEST_0_ADDRESSID=...
&PAYMENTREQUEST_0_ADDRESSSTATUS=Confirmed
```

6. Invoke a form on your site that invokes the `DoExpressCheckoutPayment` API operation on the Sandbox:

```
<form method=post action=https://api-3t.sandbox.paypal.com/nvp>
   <input type=hidden name=USER value=API_username>
   <input type=hidden name=PWD value=API_password>
   <input type=hidden name=SIGNATURE value=API_signature>
   <input type=hidden name=VERSION value=XX.0>
   <input type=hidden name=PAYMENTREQUEST_0_PAYMENTACTION
      value=Authorization>
   <input type=hidden name=PAYERID value=7AKUSARZ7SAT8>
   <input type=hidden name=TOKEN value= EC%2d1NK66318YB717835M>
   <input type=hidden name=PAYMENTREQUEST_0_AMT value= 19.95>
   <input type=submit name=METHOD value=DoExpressCheckoutPayment>
</form>
```

7. Review the response string from the `DoExpressCheckoutPayment` API operation.

   If the operation was successful, the response should include `ACK` set to `Success`, as follows:

```
TIMESTAMP=2007%2d04%2d05T23%3a30%3a16Z
&CORRELATIONID=333fb808bb23
ACK=Success
&VERSION=XX%2e000000
&BUILD=1%2e0006
&TOKEN=EC%2d1NK66318YB717835M
&PAYMENTREQUEST_0_TRANSACTIONID=043144440L487742J
&PAYMENTREQUEST_0_TRANSACTIONTYPE=expresscheckout
&PAYMENTREQUEST_0_PAYMENTTYPE=instant
&PAYMENTREQUEST_0_ORDERTIME=2007%2d04%2d05T23%3a30%3a14Z
&PAYMENTREQUEST_0_AMT=19%2e95
&PAYMENTREQUEST_0_CURRENCYCODE=USD
&PAYMENTREQUEST_0_TAXAMT=0%2e00
&PAYMENTREQUEST_0_PAYMENTSTATUS=Pending
&PAYMENTREQUEST_0_PENDINGREASON=authorization
&PAYMENTREQUEST_0_REASONCODE=None
```

Handling Payment Settlements With Express Checkout

You can use PayPal API operations to handle the capture of payments authorized using Express Checkout and manage Express Checkout authorization and order payment actions.

- Sale Payment Action for Express Checkout
- Authorization Payment Action for Express Checkout
- Order Payment Action for Express Checkout

Sale Payment Action for Express Checkout

A *sale* payment action represents a single payment that completes a purchase for a specified amount.

A sale is the default Express Checkout payment action; however, you can also specify the following action in your `SetExpressCheckout` and `DoExpressCheckoutPayment` requests:

```
PAYMENTREQUEST_n_PAYMENTACTION=Sale
```

A sale is the most straightforward payment action. Choose this payment action if the transaction, including shipping of goods, can be completed immediately. To use this payment action:

- The final amount of the payment must be known when you invoke the `DoExpressCheckoutPayment` API operation
- You should intend to fulfill the order immediately, such as would be the case for digital goods or for items you have in stock for immediate shipment

After you execute the `DoExpressCheckoutPayment` API operation, the payment is complete and further action is unnecessary. You cannot capture a further payment or void any part of the payment when you use this payment action.

Authorization Payment Action for Express Checkout

An *authorization* payment action represents an agreement to pay and places the buyer's funds on hold for up to three days.

To set up an authorization, specify the following payment action in your `SetExpressCheckout` and `DoExpressCheckoutPayment` requests:

```
PAYMENTREQUEST_n_PAYMENTACTION=Authorization
```

An authorization enables you to capture multiple payments up to 115% of, or USD $75 more than, the amount you specify in the `DoExpressCheckoutPayment` request. Choose this payment action if you need to ship the goods before capturing the payment or if there is some reason not to accept the payment immediately.

The *honor period*, for which funds can be held, is three days. The *valid period*, for which the authorization is valid, is 29 days. You can reauthorize the 3-day honor period at most once within the 29-day valid period.

You can void an authorization, in which case, the uncaptured part of the amount specified in the `DoExpressCheckoutPayment` request becomes void and can no longer be captured. If no part of the payment has been captured, the entire payment becomes void and nothing can be captured.

API operations associated with Authorization payment action in Express Checkout

API Operation	Description
`DoCapture`	Capture an authorized payment
`DoReauthorization`	Reauthorize a payment
`DoVoid`	Void an order or an authorization

Order Payment Action for Express Checkout

An *order* payment action represents an agreement to pay one or more authorized amounts up to the specified total over a maximum of 29 days.

To set up an order, specify the following payment action in your `SetExpressCheckout` and `DoExpressCheckoutPayment` requests:

```
PAYMENTREQUEST_n_PAYMENTACTION=Order
```

An order enables you to create multiple authorizations over the 29 days; each authorization you create places the buyer’s funds on hold for up to three days. You can capture multiple payments for each authorization, up to 115% of, or USD $75 more than, the amount you specify in the `DoExpressCheckoutPayment` request.

NOTE: The default number of child authorizations in your PayPal account is 1. To do multiple authorizations please contact PayPal to request an increase.

This payment action provides the most flexibility and should be used when either a sale or one authorization plus one reauthorization do not meet your needs. Situations in which orders are appropriate include the handling of

- Back orders, in which available merchandise is sent immediately and the remaining merchandise is sent when available, which may include more than two shipments
- Split orders, in which merchandise is sent in more than one shipment, perhaps to different addresses, and you want to collect a payment for each shipment
- Drop shipments, which are shipments from other vendors for which you accept the payment

You cannot reauthorize an authorization. You handle the need to reauthorize, for example when the hold period or valid period of an authorization expires, simply by creating another authorization.

You can void an order or an authorization created from the order. If you void an order, the uncaptured part of the amount specified in the `DoExpressCheckoutPayment` request becomes void and can no longer be captured. If no part of the payment has been captured, the entire payment becomes void and nothing can be captured.

If you void an authorization associated with the order, the uncaptured part of the amount specified in the authorization becomes void and can no longer be captured. If no part of the authorization has been captured, the entire authorized payment becomes void.

API operations associated with Order payment action in Express Checkout

API Operation	Description
`DoAuthorization`	Authorize a payment
`DoCapture`	Capture an authorized payment
`DoVoid`	Void an order or an authorization

Issuing Refunds

You can use the `RefundTransaction` PayPal API operation to issue refunds.

Use the `RefundTransaction` API to issue one or more refunds associated with a transaction, such as a transaction created by a capture of a payment. The transaction is identified by a transaction ID that PayPal assigns when the payment is captured.

NOTE: You cannot make a refund if the transaction occurred after the refund period has passed; typically, the refund period is 60 days.

You can refund amounts up to the total amount of the original transaction. If you specify a a full refund, the entire amount is refunded. If you specify a partial refund, you must specify the amount to refund, the currency, and a description of the refund, which is called a *memo*.

When you call the `RefundTransaction` API, PayPal responds with another transaction ID, which is associated with the refund (not the original transaction), and additional information about the refund. This information identifies

- the gross amount of the refund, which is returned to the payer
- the amount of the refund associated with the original transaction fee, which is returned to you
- the net amount of the refund, which is deducted from your balance

To issue a refund

1. In the `RefundTransaction` request, specify the transaction ID of the transaction whose payment you want to refund.

   TRANSACTIONID=*transaction_id*

2. Specify the kind of refund, which is either `Full` or `Partial`.

```
REFUNDTYPE=Full
```

or

```
REFUNDTYPE=Partial
```

3. For a partial refund, specify the refund amount, including the currency.

```
AMT=amount
CURRENCYCODE=currencyID
```

4. For a partial refund, specify the memo description.

```
NOTE=description
```

5. Execute the `RefundTransaction` operation.
6. Check the acknowledgement status in the `RefundTransaction` response to ensure that the operation was successful.

6 Integrating Recurring Payments

Both Direct Payment and Express Checkout support recurring payments. You can set up a recurring payment to handle a subscription and other payments that occur on a fixed schedule.

How Recurring Payments Work

When you support recurring payments for a buyer, you create a *recurring payments profile*. The profile contains information about the recurring payments, including details for an optional trial period and a regular payment period. Both periods contain information about the payment frequency and payment amounts, including shipping and tax, if applicable.

After a profile is created, PayPal automatically queues payments based on the billing start date, billing frequency, and billing amount. Payments reoccur until the profile expires, there are too many failed payments to continue, or you cancel the profile.

NOTE: When using Express Checkout, the buyer can also cancel a recurring payments profile.

When a buyer uses Express Checkout, queued payments are funded using the normal funding source hierarchy within the buyer’s PayPal account.

After you create a recurring payments profile, you can view recurring payments details or cancel the recurring payments profile from your PayPal account.You can also access recurring payments reports from the PayPal **Business Overview** page.

Also, after creating a recurring payments profile, you can use the Recurring Payments API to do the following:

- Get information details about a recurring payments profile.
- Change the status of a recurring payments profile.

- Update the details of the recurring payments profile.
- Bill the outstanding amount of the recurring payments profile.

Limitations

The current release of the Recurring Payments API has the following limitations:

- A profile can have at most one optional trial period and a single regular payment period.
- The profile start date may not be earlier than the profile creation date.

Recurring payments with Express Checkout also has the following limitations:

- To be able to create a recurring payments profile for the buyer, the buyer's PayPal account must include an active credit card.
- At most ten recurring payments profiles can be created during checkout.
- You can only increase the profile amount by 20% in each 180-day interval after the profile is created.

Recurring Payments Terms

Some terms are commonly used by PayPal in the context of recurring payments.

Recurring payments terms

Term	Definition
Recurring payments profile	Your record of a recurring transaction for a single buyer. The profile includes all information required to automatically bill the buyer a fixed amount of money at a fixed interval.
Billing cycle	One payment is made per billing cycle. Each billing cycle is made up of two components. • The billing period specifies the unit to be used to calculate the billing cycle (such as days or months). • The billing frequency specifies the number of billing periods that make up the billing cycle. For example, if the billing period is Month and the billing frequency is 2, the billing cycle will be two months. If the billing period is Week and the billing frequency is 6, the payments will be scheduled every 6 weeks.
Regular payment period	The main subscription period for this profile, which defines a payment amount for each billing cycle. The regular payment period begins after the trial period, if a trial period is specified for the profile.
Trial period	An optional subscription period before the regular payment period begins. A trial period may not have the same billing cycles and payment amounts as the regular payment period.

Term	Definition
Payment amount	The amount to be paid by the buyer for each billing cycle.
Outstanding balance	If a payment fails for any reason, that amount is added to the profile's outstanding balance.
Profile ID	An alphanumeric string (generated by PayPal) that uniquely identifies a recurring profile. You can specify the Profile ID in the `TransactionSearch` API operation to obtain all payments associated with the identified profile.

Recurring Payments With Direct Payment

Recurring payments with Direct Payment enables a recurring payment to be associated with a debit or credit card. For these payments, you must collect on your website all necessary information from your buyer, including billing amount and buyer's credit card information.

After you have collected the information, call the `CreateRecurringPaymentsProfile` API for each profile to be created. The `CreateRecurringPaymentsProfile` request must contain all required credit card information and must not contain a value for the `TOKEN` field.

The following table lists the fields that are required in the `CreateRecurringPaymentsProfile` request for recurring payments using direct payments.

Required fields for CreateRecurringPaymentsProfile in Direct Payment

NVP	SOAP
`CREDITCARDTYPE`	`CreditCardDetails.CreditCardType`
`ACCT`	`CreditCardDetails.CreditCardNumber`
`EXPDATE`	`CreditCardDetails.ExpMonth` and `CreditCardDetails.ExpYear`
`FIRSTNAME`	`CreditCardDetails.CardOwner.PayerName.FirstName`
`LASTNAME`	`CreditCardDetails.CardOwner.PayerName.LastName`
`PROFILESTARTDATE`	`RecurringPaymentProfileDetails.BillingStartDate`
`BILLINGPERIOD`	`ScheduleDetails.PaymentPeriod.BillingPeriod`
`BILLINGFREQUENCY`	`ScheduleDetails.PaymentPeriod.BillingFrequency`
`AMT`	`ScheduleDetails.PaymentsPeriod.Amount`

The `CreateRecurringPaymentsProfile` response contains a Profile ID, which is an encoded string that uniquely identifies the recurring payments profile.

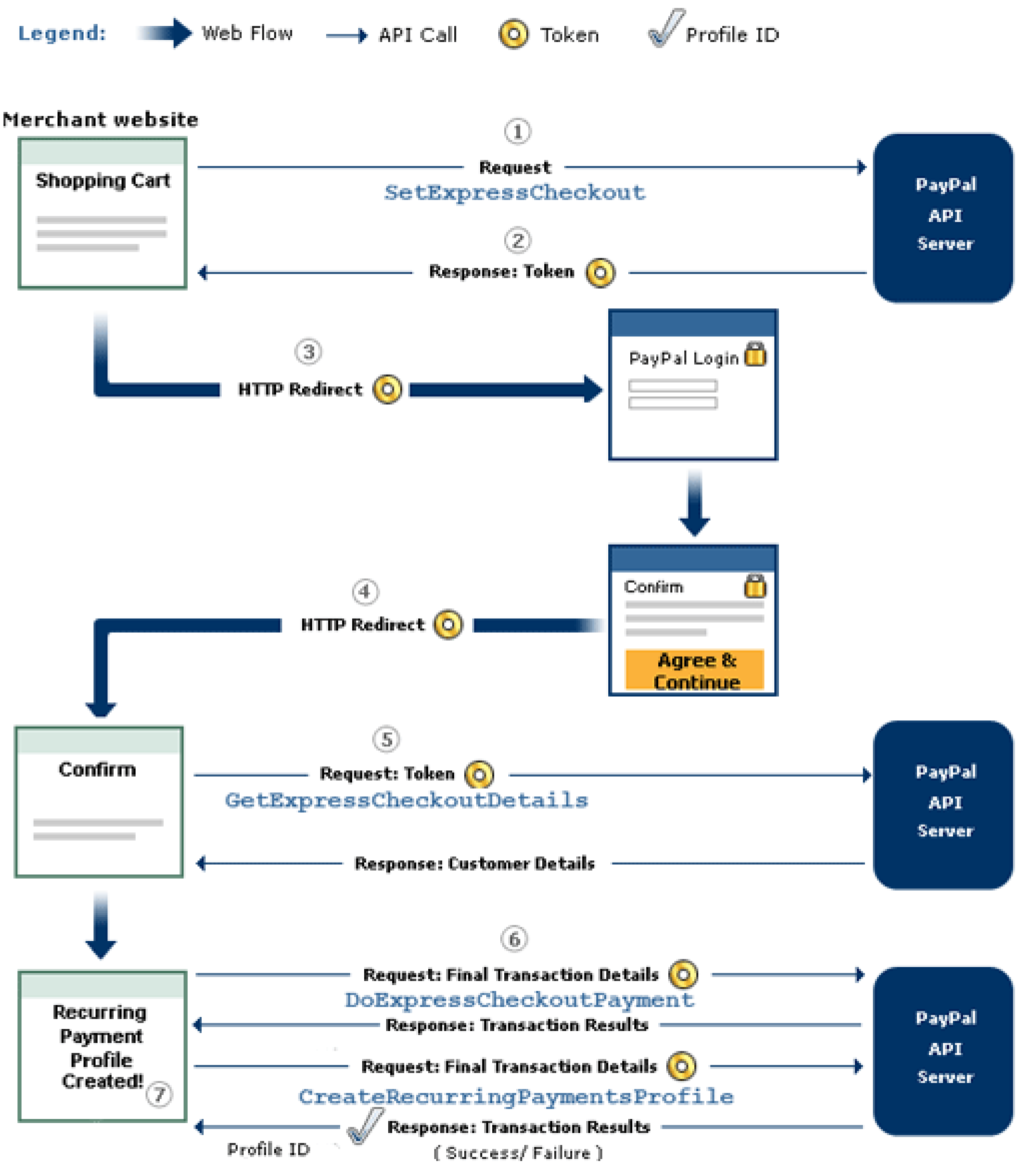
For more options when creating a recurring payments profile, see "Options for Creating a Recurring Payments Profile" on page 66.

As with all direct payments, PayPal is completely invisible to your buyer before, during, and after the purchase. PayPal does not send an email receipt to the buyer, nor will the buyer's credit card statement indicate that PayPal processed the payment.

Recurring Payments With Express Checkout

During the Express Checkout flow, you can create one or more recurring payments and mix recurring payments with other purchases.

The following diagram illustrates the typical processing flow to create recurring payments during checkout.



The circled numbers in the diagram correspond to the following steps:

Recurring payments processing flow

Step	Merchant...	PayPal...
1	Calls `SetExpressCheckout` with one or more billing agreement details in the request	
2		Returns a token, which identifies the transaction, to the merchant.
3	Redirects buyer's browser to: `https://www.paypal.com/cgi-bin/webscr?cmd=_express-checkout&token=<`*`token returned by`* `SetExpressCheckout>`	
		Displays login page. Allows user to select payment options and shipping address.
4		Redirects buyer's browser to `returnURL` passed to `SetExpressCheckout` if buyer agrees to payment description.
5	Calls `GetExpressCheckoutDetails` to get buyer information (optional).	
		Returns `GetExpressCheckoutDetails` response.
	Displays merchant review page for buyer.	
6	Calls `DoExpressCheckoutPayment` if the order includes one-time purchases as well as a recurring payment. Otherwise, skip this step.	
		Returns `DoExpressCheckoutPayment` response
	Calls `CreateRecurringPaymentsProfile` one time for each recurring payment item included in the order.	
		Returns `ProfileID` in `CreateRecurringPaymentsProfile` response for each profile successfully created.
7	Displays successful transaction page.	

Initiating the Processing Flow With SetExpressCheckout

As in the Express Checkout flow, the `SetExpressCheckout` request notifies PayPal that you are:

- Initiating an order that can be either a one-time purchase, up to ten recurring payments, or a mixture of a one-time purchase and recurring payments
- Initiating the processing flow to create one or more billing agreements for recurring payments with no associated one-time purchase or recurring payment

NOTE: You can also initiate the processing flow using `SetCustomerBillingAgreement` for orders that contain only a single recurring payment.

Typically, you set the amount of the payment for an Express Checkout transaction when you call the `SetExpressCheckout` request and confirm the amount in the `DoExpressCheckoutPayment` request. If, however, you set the amount of the payment to 0 in the `SetExpressCheckout` request, you can create a billing agreement without initiating a payment.

NOTE: To create a billing agreement without purchase, use Version 54.0 or higher, of the PayPal API.

To set up one or more billing agreements for recurring payments, modify the `SetExpressCheckout` request as follows:

1. Add an `L_BILLINGTYPE`*n* field for each billing agreement you want to create; *n* is a value in the range of `0` to `9`, inclusive. Set the value of each field to `RecurringPayments`.

   ```
   L_BILLINGTYPE0=RecurringPayments
   ```

2. Add an `L_BILLINGAGREEMENTDESCRIPTION`*n* field to correspond to each `L_BILLINGTYPE`*n* field you pass; *n* is a value in the range of `0` to `9`, inclusive. Set the value of each field to the description of the goods or services associated with that billing agreement, for example:

   ```
   L_BILLINGAGREEMENTDESCRIPTION0=Time Magazine subscription
   ```

3. If there is no associated purchase, set `AMT` to `0`.

   ```
   AMT=0
   ```

4. (Optional) Set `MAXAMT` to the average expected transaction amount.

   PayPal uses the value you pass to validate the buyer's funding source for recurring payments. If you do not specify a value, the default is 25.00.

NOTE: When creating the recurring payments profile, pass the same `L_BILLINGTYPE`*n* value and `L_BILLINGAGREEMENTDESCRIPTION`*n* string describing the goods or service in the call to `CreateRecurringPaymentsProfile` that you passed in the call to `SetExpressCheckout`.

The `SetExpressCheckout` response provides a token that uniquely identifies the transaction for subsequent redirects and API calls.

Redirecting the Buyer’s Browser to PayPal

After you receive a successful response from `SetExpressCheckout`, add the `TOKEN` from the `SetExpressCheckout` response as a name/value pair to the following URL, and redirect your buyer’s browser to it:

```
https://www.paypal.com/cgi-bin/webscr?cmd=_express-checkout&
token=<value_from_SetExpressCheckoutResponse>
```

For redirecting the buyer’s browser to the PayPal login page, PayPal recommends that you use the HTTPS response 302 “Object Moved” with the URL above as the value of the `Location` header in the HTTPS response. Ensure that you use an SSL-enabled server to prevent browser warnings about a mix of secure and insecure graphics.

Getting Buyer Details Using GetExpressCheckoutDetails

The `GetExpressCheckoutDetails` method returns information about the buyer, including name and email address stored on PayPal. You can optionally call this API after PayPal redirects the buyer’s browser to the `ReturnURL` you specified in the `SetExpressCheckout` request.

The `GetExpressCheckoutDetails` request has one required parameter, `TOKEN`, which is the value returned in the `SetExpressCheckout` response.

The values you specified for the following parameter fields are not returned in the `GetExpressCheckoutDetails` response unless the transaction includes a purchase. The fields are ignored if you set up a billing agreement for a recurring payment that is not immediately charged.

- `DESC`
- `CUSTOM`
- `INVNUM`

Creating the Profiles With CreateRecurringPaymentsProfile

After your buyer has agreed to the recurring payments billing agreement on your confirmation page, you must call `CreateRecurringPaymentsProfile` to create the profile. If you are creating multiple recurring payments profiles, you must call `CreateRecurringPaymentsProfile` once for each profile to be created.

If the transaction includes a mixture of a one-time purchase and recurring payments profiles, call `DoExpressCheckoutPayment` to complete the one-time purchase transaction, and then call `CreateRecurringPaymentsProfile` for each recurring payment profile to be created.

IMPORTANT: The recurring payments profile is not created until you receive a success response from the `CreateRecurringPaymentsProfile` call.

The `CreateRecurringPaymentsProfile` response contains a Profile ID, which is an encoded string that uniquely identifies the recurring payments profile.

Options for Creating a Recurring Payments Profile

You can create a recurring payments profile that allows a regular payment period, an optional trial period, an initial payment, and other options.

Specifying the Regular Payment Period

Each recurring payments profile has a regular payment period that defines the amount and frequency of the payment. The following table lists the required fields for specifying the regular payment period.

Required fields for specifying a regular payment period

NVP	SOAP	Description
`PROFILESTARTDATE`	`RecurringPaymentsProfileDetails.BillingStartDate`	The date when billing for this profile begins. **NOTE:** The profile may take up to 24 hours for activation.
`BILLINGPERIOD`	`ScheduleDetails.PaymentPeriod.BillingPeriod`	The unit of measure for the billing cycle. Must be one of: ● `Day` ● `Week` ● `SemiMonth` ● `Month` ● `Year`
`BILLINGFREQUENCY`	`ScheduleDetails.PaymentPeriod.BillingFrequency`	Number of billing periods that make up one billing cycle. **NOTE:** The combination of billing frequency and billing period must be less than or equal to one year. **NOTE:** If the billing period is `SemiMonth`., the billing frequency must be 1.
`AMT`	`ScheduleDetails.PaymentPeriod.Amount`	Amount to bill for each billing cycle.

You can optionally include a value for `TOTALBILLINGCYCLES` (SOAP field `ScheduleDetails.PaymentPeriod.TotalBillingCycles`), which specifies the total number of billing cycles in the regular payment period. If no value is specified or if the value is 0, the payments continue until the profile is canceled or suspended. If the value is greater than 0, the regular payment period will continue for the specified number of billing cycles.

You can also specify an optional shipping amount or tax amount for the regular payment period.

Including an Optional Trial Period

You can optionally include a trial period in the profile by specifying the following fields in the `CreateRecurringPaymentsProfile` request. The following table lists the required fields for creating an optional trial period.

Required fields for specifying a trial period

NVP	SOAP
`TRIALBILLINGPERIOD`	`ScheduleDetails.TrialPeriod.BillingPeriod`
`TRIALBILLINGFREQUENCY`	`ScheduleDetails.TrialPeriod.BillingFrequency`
`TRIALAMT`	`ScheduleDetails.TrialPeriod.Amount`
`TRIALTOTALBILLINGCYCLES`	`ScheduleDetails.TrialPeriod.TotalBillingCycles`

Specifying an Initial Payment

You can optionally specify an initial non-recurring payment when the recurring payments profile is created by including the following fields in the `CreateRecurringPaymentsProfile` request:

Required fields for specifying an initial payment

NVP	SOAP
`INITAMT`	`ScheduleDetails.ActivationDetails.InitialAmount`
`FAILEDINITAMTACTION`	`ScheduleDetails.ActivationDetails.FailedInitAmountAction`

By default, PayPal will not activate the profile if the initial payment amount fails. You can override this default behavior by setting the `FAILEDINITAMTACTION` field to `ContinueOnFailure`, which indicates that if the initial payment amount fails, PayPal should add the failed payment amount to the outstanding balance due on this recurring payment profile.

If this field is not set or is set to `CancelOnFailure`, PayPal will create the recurring payment profile, but will place it into a pending status until the initial payment is completed. If the initial payment clears, PayPal will notify you by IPN that the pending profile has been activated. If the payment fails, PayPal will notify you by IPN that the pending profile has been canceled.

The buyer will receive an email stating that the initial payment cleared or that the pending profile has been canceled if the profile was created using Express Checkout.

Maximum Number of Failed Payments

By including the `MAXFAILEDPAYMENTS` field in the `CreateRecurringPaymentsProfile` request, you set the number of failed payments allowed before the profile is automatically suspended. You receive an IPN message when the number of failed payments reaches the maximum number specified.

Billing the Outstanding Amount

If a payment fails due to any reason, the amount that was to be billed (including shipping and tax, if applicable) is added to the profile's outstanding balance. Use the `AUTOBILLOUTAMT` field in the `CreateRecurringPaymentsProfile` request to specify whether or not the outstanding amount should be added to the payment amount for the next billing cycle.

Whether or not you choose to include the outstanding amount with the payment for the next billing cycle, you can also use the `BillOutstandingAmount` API to programmatically collect that amount at any time.

Recurring Payments Profile Status

The recurring payments actions you may take, depend on the status of the profile.

A recurring payments profile can have one of the following status values:

- `ActiveProfile`
- `PendingProfile`
- `ExpiredProfile`
- `SuspendedProfile`
- `CancelledProfile`

If the profile is successfully created, it has an `ActiveProfile` status. However, if a non-recurring initial payment fails and `FAILEDINITAMTACTION` is set to `CancelOnFailure` in the `CreateRecurringPaymentsProfile` request, the profile is created with a status of `PendingProfile` until the initial payment either completes successfully or fails.

A profile has a status of `ExpiredProfile` when both the total billing cycles for both the optional trial period and the regular payment period have been completed.

You can suspend or cancel a profile by using the `ManageRecurringPaymentsProfileStatus` API. You can also reactivate a suspended profile. If the maximum number of failed payments has already been reached, however, you will need to increase the number of failed payments before reactivating the profile.

NOTE: You can also suspend, cancel, or reactive a recurring payments profile through the PayPal website.

For recurring payments profiles created with Express Checkout, the buyer receives an email about the change in status of their recurring payment.

Getting Recurring Payments Profile Information

Use the `GetRecurringPaymentsProfileDetails` API to get information about a profile.

NOTE: You can also get information about recurring payments profiles from the PayPal website.

Along with the information that you specified in the `CreateRecurringPaymentsProfile` request, `GetRecurringPaymentsProfileDetails` also returns the following summary information about the profile:

- Profile status
- Next scheduled billing date
- Number of billing cycles completed in the active subscription period
- Number of billing cycles remaining in the active subscription period
- Current outstanding balance
- Total number of failed billing cycles
- Date of the last successful payment received
- Amount of the last successful payment received

Modifying a Recurring Payments Profile

Use the `UpdateRecurringPaymentsProfile` API to modify a recurring payments profile.

NOTE: You can also modify recurring payments profiles from the PayPal website.

You can only modify the following specific information about an active or suspended profile:

- Subscriber name or address
- Past due or outstanding amount
- Whether to bill the outstanding amount with the next billing cycle
- Maximum number of failed payments allowed
- Profile description and reference
- Number of additional billing cycles
- Billing amount, tax amount, or shipping amount

NOTE: You cannot modify the billing frequency or billing period of a profile. You can modify the number of billing cycles in the profile.

NOTE: For recurring payments with Express Checkout, certain updates, such as billing amount, are not allowed within 3 days of the scheduled billing date, and an error is returned.

You can modify the following profile information during the trial period or regular payment period.

- Billing amount
- Number of billing cycles

The profile changes take effect with the next payment after the call to update the profile. Say, for example, the buyer has made one trial payment out of a total of three. `UpdateRecurringPaymentsProfile` is called to increase the number of billing cycles to five. As a result, the buyer will have four additional trial payments to make. If the call to `UpdateRecurringPaymentsProfile` is made during the regular payment period, the changes take effect with the buyer's next scheduled regular payment.

For complete details, see the *Name-Value Pair Developer Guide and Reference* or the *SOAP API Reference*.

Updating Addresses

When you update the subscriber shipping address, you must enter all of address fields, not just those that are changing:

For example, if you want to update the subscriber's street address, you must specify all of the address fields listed in the *Name-Value Pair Developer Guide and Reference* or *SOAP API Reference*, not just the field for the street address.

Updating the Billing Amount

For profiles created using Express Checkout, the total amount of a recurring payment can only be increased 20% in a fixed 180-day interval after the profile is created. The 20% maximum is based on the total amount of the profile at the beginning of the 180-day interval, including any shipping or tax amount.

For example, if a profile is created on March 10 with a total amount of $100, then during the 180-day interval from March 10 to September 6, you can increase the payment amount to a maximum of $120 (120% of $100).

Suppose that during the first 180-day interval, you increased the payment amount to $110. Then during the next 180-day interval (starting on September 7 in this example), you can only increase the amount of the payment to a maximum of $132 (120% of $110).

Billing the Outstanding Amount of a Profile

Use the `BillOutstandingAmount` API to immediately bill the buyer for the current past due or outstanding amount for a recurring payments profile.

NOTE: You can also bill the buyer for the current past due or outstanding amount for a recurring payments profile from the PayPal website.

To bill the outstanding amount:

- The profile status must be active or suspended.

  NOTE: The `BillOutstandingAmount` API does not reactivate a suspended profile. You need to call `ManageRecurringProfileStatus` to do this.

- The profile must have a non-zero outstanding balance.
- The amount of the payment cannot exceed the outstanding amount for the profile.
- The `BillOutstandingAmount` call cannot be within 24 hours of a regularly scheduled payment for this profile.

NOTE: If another outstanding balance payment is already queued, an API error is returned.

You will be informed by IPN about the success or failure of the outstanding payment. For profiles created using Express Checkout, the buyer will receive an email notification of the payment.

Recurring Payments Notifications

You are notified of recurring payments events through IPN and email; however, using `GetTransactionDetails` to obtain the information you need is typically sufficient.

You are notified of certain events through IPN. For recurring payments profiles created using Express Checkout, buyers are also notified of specific events by email. The following table indicates when IPN and emails are generated.

Recurring payments IPN messages and email

Event	IPN	Buyer Email
Profile successfully created	Yes	Yes
Profile creation failed	Yes	Yes
Profile canceled from paypal.com interface	Yes	Yes
Profile status changed using API	No	Yes
Profile update using API	No	Yes
Initial payment either succeeded or failed	Yes	Yes
Payment either succeeded or failed (during either trial period or regular payment period)	Yes	Yes
Outstanding payment either succeeded or failed	Yes	Yes
Maximum number of failed payments reached	Yes	No

NOTE: API transactions such as `ManangeRecurringPaymentsProfileStatus` do not trigger IPN notification because the success or failure of the call is provided immediately by the API response.

7 Getting Started With the PayPal Name-Value Pair API

The PayPal Name-Value Pair (NVP) API provides parameter-based association between request and response fields of a message and their values. The request message is sent via the API from your website and a response message is returned by PayPal using a client-server model in which your site is a client of the PayPal server.

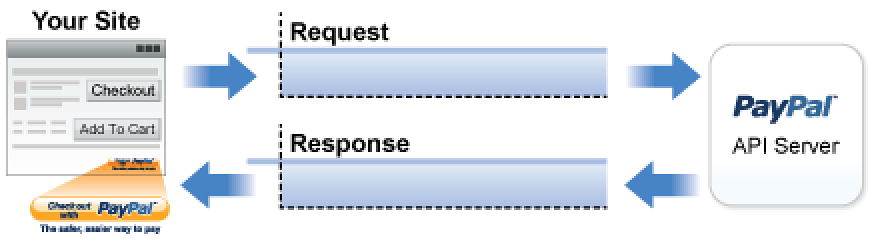
NOTE: The PayFlow API also uses name-value pairs to provide parameter-based association between request and response fields of a message and their values; however, the PayFlow API is not the same as the NVP API; for more information about the PayFlow API, see Website Payments Pro Payflow Edition Developer Guide.

- PayPal API Client-Server Architecture
- Obtaining API Credentials
- Creating an NVP Request
- Executing NVP API Operations
- Responding to an NVP Response

PayPal API Client-Server Architecture

The PayPal API uses a client-server model in which your website is a client of the PayPal server.

A page on your website initiates an action on a PayPal API server by sending a request to the server. The PayPal server responds with a confirmation that the requested action was taken or or indicates that an error occurred. The response might also contain additional information related to the request. The following diagram shows the basic request-response mechanism.

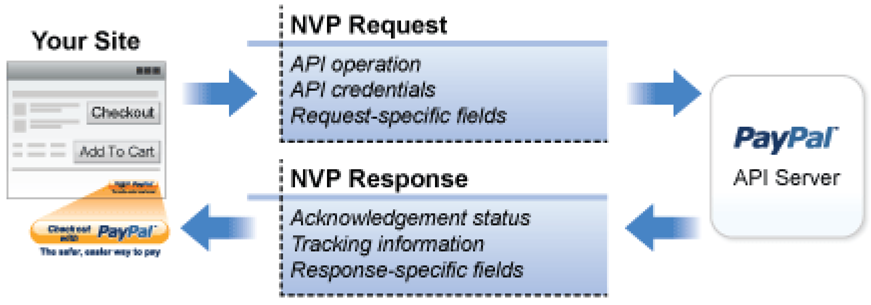


For example, you might want to obtain the buyer’s shipping address from PayPal. You can initiate a request specifying an API operation that gets buyer details. The response from the PayPal API server contains information about whether the request was successful. If the operation succeeds, the response contains the requested information. In this case, the response contains the buyer’s shipping address. If the operation fails, the response contains one or more error messages.

PayPal Name-Value Pair API Requests and Responses

To perform a PayPal NVP API operation, you send an NVP-formatted request to a PayPal NVP server and interpret the response.

In the following diagram, your website generates a request. The request is executed on a PayPal server and the response is returned to your site.



The request identifies

- The name of the API operation to be performed and its version
- Credentials that identify the PayPal account making the request
- Request-specific information that controls the API operation to be performed

A PayPal API server performs the operation and returns a response. The response contains

- An acknowledgement status that indicates whether the operation was a success or failure and whether any warning messages were returned
- Information that can be used by PayPal to track execution of the API operation
- Response-specific information required to fulfill the request

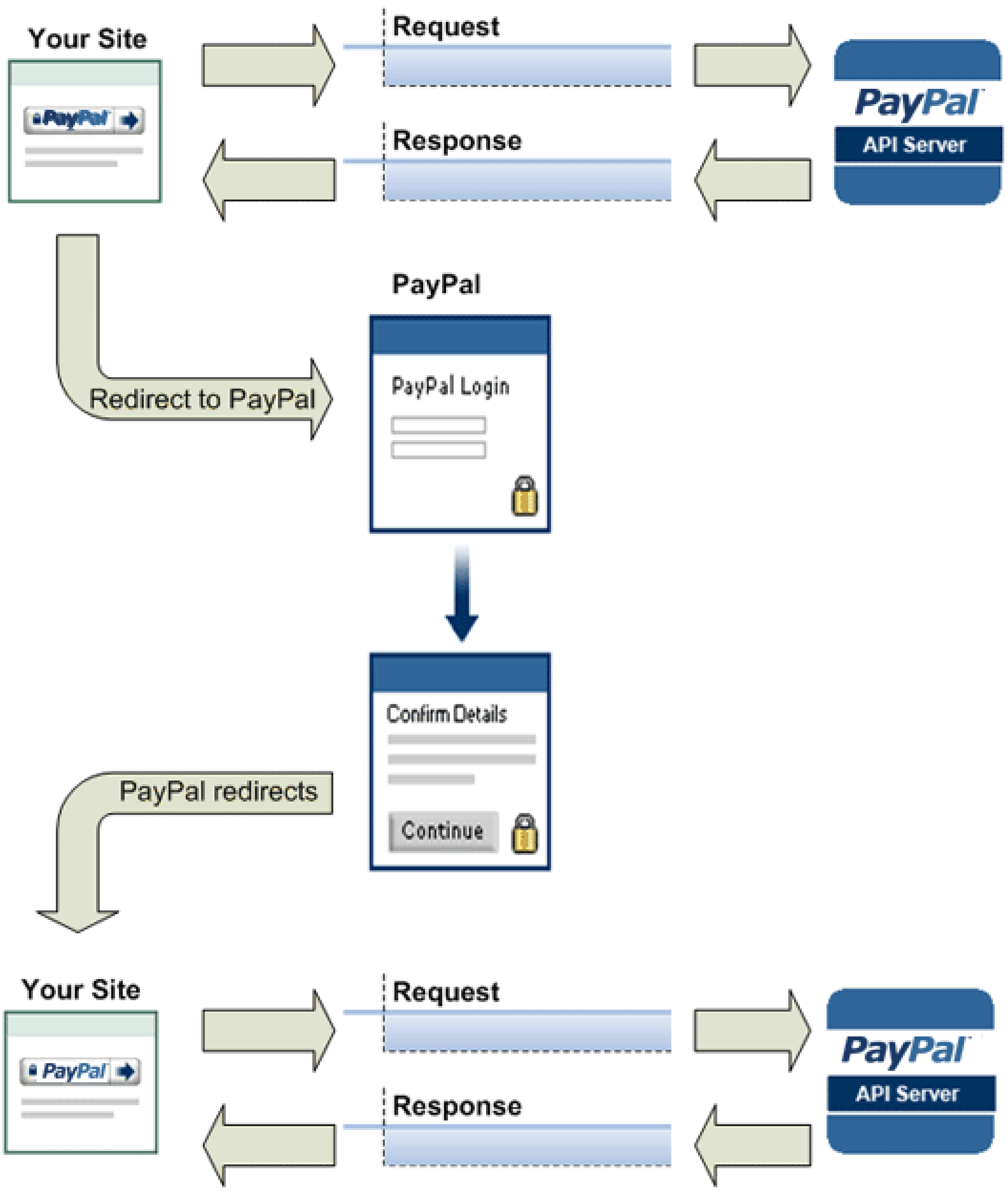
Multiple API Operations

Some of the features, such as Express Checkout, require you to call multiple API operations.

Typically, these features require you to

1. Invoke an API operation, such as `SetExpressCheckout`, that sets up the return URL to which PayPal redirects your buyer's browser after the buyer finishes on PayPal. Other setup also can be performed by this API operation.
2. Invoke additional API operations after receiving the buyer's permission on PayPal, for example, `GetExpressCheckoutDetails` or `DoExpressCheckoutPayment`.

The following diagram shows the execution flow between your site and PayPal:



Token Usage

Typically, the API operation that sets up a redirection to PayPal returns a token. This token is passed as a parameter in the redirect to PayPal. The token also might be required in related API operations.

Obtaining API Credentials

To use the PayPal API, you must have API credentials that identify you as a PayPal Business account holder who is authorized to perform various API operations. Although you can use either an API *signature* or a *certificate* for credentials, PayPal recommends you use a signature.

IMPORTANT: Although you can have both a signature and certificate, you cannot use both at the same time.

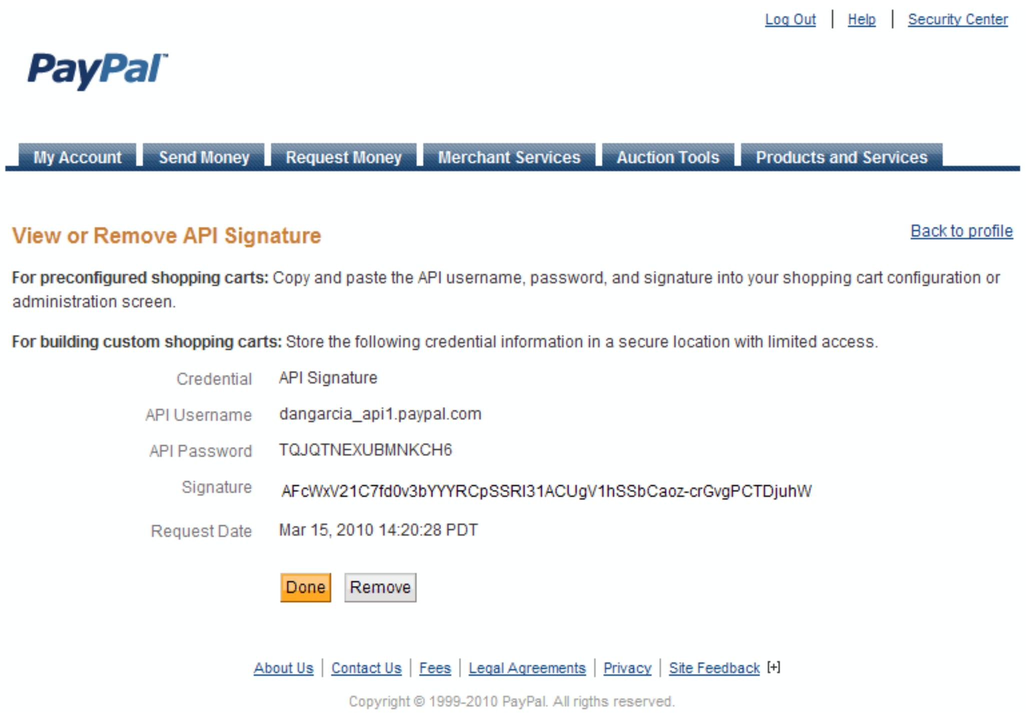
Creating an API Signature

An API signature consists of an API username along with an associated API password and signature, all of which are assigned by PayPal. You need to include this information whenever you execute a PayPal API operation.

You must have a PayPal Business account to create a signature.

To create an API signature:

1. Log into PayPal, then click **Profile** under **My Account**.
2. Click **API Access**.
3. Click **Request API Credentials**.
4. Check **Request API signature** and click **Agree and Submit**.



View or Remove API Signature

For preconfigured shopping carts: Copy and paste the API username, password, and signature into your shopping cart configuration or administration screen.

For building custom shopping carts: Store the following credential information in a secure location with limited access.

Credential	API Signature
API Username	dangarcia_api1.paypal.com
API Password	TQJQTNEXUBMNKCH6
Signature	AFcWxV21C7fd0v3bYYYRCpSSRl31ACUgV1hSSbCaoz-crGvgPCTDjuhW
Request Date	Mar 15, 2010 14:20:28 PDT

Copyright © 1999-2010 PayPal. All rigths reserved.

5. Click **Done to complete the process**.

Creating an API Certificate

Create an API certificate only if your website requires it. Typically, you want to create an API signature for your credentials instead.

If you do need a certificate, follow the instructions at https://www.paypal.com/IntegrationCenter/ic_api-certificate.html.

NOTE: The certificate for API credentials is not the same as an SSL certificate for your website; they are not related to each other.

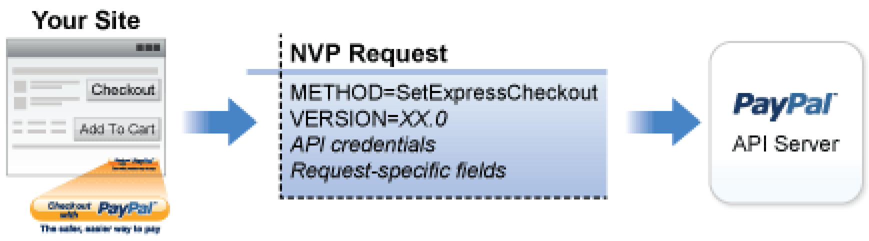
Creating an NVP Request

The Name-Value Pair request format specifies the API operation to perform, credentials that authorize PayPal to access your account, and fields containing additional information to be used in the request.

Specifying the PayPal API Operation

For the NVP version of the PayPal API, you must specify the name of the PayPal API operation to execute in each request along with the version of the API operation.

The following diagram shows the API operation part of an NVP request:



A *method* specifies the PayPal operation you want to execute and each method is associated with a *version*. Together, the method and version define the exact behavior of the API operation. Typically, the behavior of an API operation does not change between versions; however, you should carefully retest your code whenever you change a version.

To specify a method and version number:

1. Choose the PayPal API operation you want to use.

`METHOD=`*operation*

2. Choose the appropriate version.

In most cases, you should use the latest version of the API operation.

`VERSION=`*version_number*

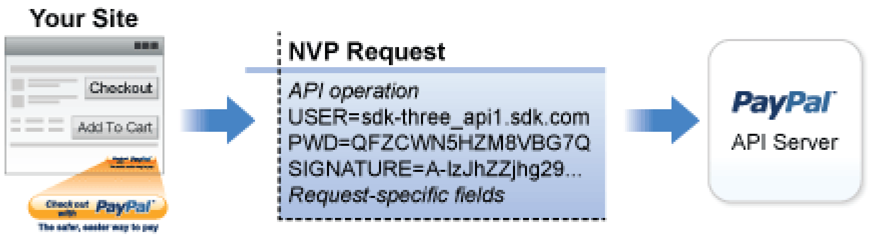
Example of setting the API operation and version using PHP

```
function PPHttpPost($methodName_, $nvpStr_) {
  ...
  $version = urlencode('XX.0');  // NVPRequest for submitting to server
  $nvpreq ="METHOD=$methodName_&VERSION=$version...$nvpStr_";
  ...
}
```

Specifying an API Credential

You must specify API credentials in each request to execute a PayPal API operation.

When you execute a PayPal API operation, you use credentials, such as a signature, to authenticate that you are requesting the API operation. The following diagram shows the API credentials part of an NVP request:



To enable PayPal to authenticate your request

1. Specify the API user name associated with your account.

USER=*API_username*

2. Specify the password associated with the API user name.

PWD=*API_password*

3. If you are using an API signature and not an API certificate, specify the API signature associated with the API username.

SIGNATURE=*API_signature*

Specifying Credentials using cURL

The following example shows one way to specify a signature using cURL:

```
curl --insecure https://api-3t.sandbox.paypal.com/nvp -d ^
"METHOD=DoDirectPayment^
&VERSION=XX.0^
&USER=API_username^
&PWD=API_password^
&SIGNATURE=API_signature^
&..."
```

NOTE: This example does not establish a secure connection and should not be used live on paypal.com.

URL Encoding

All requests to execute PayPal API operations sent via HTTP must be URL encoded.

The PayPal NVP API uses the HTTP protocol to send requests and receive responses from a PayPal API server. You must encode all data sent using the HTTP protocol because data that is not encoded could be misinterpreted as part of the HTTP protocol instead of part of the request. Most programming languages provide a way to encode strings in this way. You should consistently URL encode the complete API request; otherwise, you may find that unanticipated data causes an error.

NOTE: An HTTP form is automatically URL encoded by most browsers.

List Syntax for Name-Value Pairs

The PayPal API uses a special syntax for NVP fields defined as lists.

The NVP interface to the PayPal API requires a unique name for each field. In the API, lists are prefixed by `L_`. To identify an element within the list, use the offset from the beginning of the list, starting with 0 as the first element. For example, `L_DESC0` is the first line of a description, `L_DESC1`, is the second line, and so on.

NOTE: Not all lists follow the `L_` prefix convention; however, all lists start with 0 as the first element.

Executing NVP API Operations

You execute an PayPal NVP API operation by submitting an HTTP POST request to a PayPal API server.

Specifying a PayPal Server

You execute a PayPal API operation by submitting the request to a PayPal API server.

To execute a PayPal NVP API operation, submit your complete request to one of the following end points:

Server end point	Description
`https://api-3t.sandbox.paypal.com/nvp`	Sandbox server for use with API signatures; use for testing your API
`https://api-3t.paypal.com/nvp`	PayPal “live” production server for use with API signatures
`https://api.sandbox.paypal.com/nvp`	Sandbox server for use with API certificates; use for testing your API
`https://api.paypal.com/nvp`	PayPal “live” production server for use with API certificates

NOTE: You must use different API credentials for each server end point. Typically, you obtain API credentials when you test in the Sandbox and then obtain another set of credentials for the production server. You must change each API request to use the new credentials when you go live.

Logging API Operations

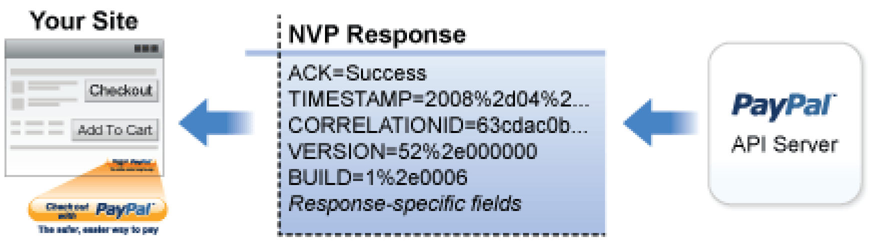
You should log basic information about each PayPal API operation you execute.

All responses to PayPal API operations contain information that may be useful for debugging purposes. You should log the Correlation ID, which identifies the API operation to PayPal, and response-specific information, such as the transaction ID, which you can use to review a transaction on the PayPal website or through the API. You can log other information that may be useful, such as the timestamp. You could implement a scheme that logs the entire request and response in a "verbose" mode; however, you should never log the password from a request.

Responding to an NVP Response

The Name-Value Pair response consists of the answer to the request as well as common fields that identify the API operation and how it was executed.

The following diagram shows fields in the response to a PayPal NVP API operation:



Common Response Fields

The PayPal API always returns common fields in addition to fields that are specific to the requested PayPal API operation.

A PayPal API response includes the following fields:

Field	Description
`ACK`	Acknowledgement status, which is one of the following values: ● `Success` indicates a successful operation. ● `SuccessWithWarning` indicates a successful operation; however, there are messages returned in the response that you should examine. ● `Failure` indicates the operation failed; the response also contains one or more error messages explaining the failure. ● `FailureWithWarning` indicates that the operation failed and that there are messages returned in the response that you should examine
`CORRELATIONID`	Correlation ID, which uniquely identifies the transaction to PayPal

Field	Description
`TIMESTAMP`	The date and time that the requested API operation was performed
`VERSION`	The version of the API
`BUILD`	The sub-version of the API

URL Decoding

All responses to HTTP POST operations used by the PayPal NVP API must be decoded.

The PayPal NVP API uses the HTTP protocol to send requests and receive responses from a PayPal API server. You must decode all data returned using the HTTP protocol so that it can be displayed properly. Most programming languages provide a way to decode strings.

NOTE: Most browsers decode responses to HTTP requests automatically.

8 Implementing 3-D Secure Transactions (UK Only)

Website Payments Pro allows UK merchants to pass 3-D Secure™ authentication data to PayPal for debit and credit cards processed with the `DoDirectPayment` API request. Updating your site with 3-D Secure enables your participation in the Verified by Visa and MasterCard SecureCode programs.

NOTE: A US or Canadian merchant can implement 3-D Secure; however, the authentication data is ignored by PayPal. This information only applies to 3-D Secure for UK merchants implementing Direct Payment.

Introduction to 3-D Secure

3-D Secure is a protocol developed by the card schemes that improves the security of Internet payments that are not within a closed-loop checkout option (such as PayPal). It allows merchants to authenticate cardholders through the cards' issuers. Its goals are to reduce the likelihood of fraud when using supported cards and to improve transaction performance. Merchants who do not use 3-D Secure may be liable for fraudulent transactions even if the transaction was authorized by other means. Visa offers 3-D Secure under the name Verified by Visa and MasterCard offers it as MasterCard SecureCode.

PayPal enables you to pass 3-D Secure data to PayPal for Payments Pro transactions, but you must obtain the 3-D Secure authentication data from the card's issuer. PayPal has an agreement with Cardinal Commerce that allows Payments Pro merchants free access to Cardinal's 3-D Secure technology, Cardinal Centinel™. The Cardinal Centinel® Thin Client interface provides access to payer authentication for transactions using Visa, MasterCard, and Maestro branded cards. Use of 3-D Secure authentication is optional for Visa and MasterCard transactions.

3-D Secure is *not supported* for direct Recurring Billing and Reference Transactions. Cards that require 3-D Secure authentication cannot use these APIs; however, cards where 3-D Secure is optional can continue to process transactions without authentication. If you use either of these features in your current integration, you must exclude the Maestro card type from the available options.

NOTE: Merchants must register with Cardinal Commerce before using this feature.

For more information, see

- Verified by Visa: http://www.visaeurope.com/personal/onlineshopping/verifiedbyvisa/main.jsp
- MasterCard SecureCode: http://www.mastercard.com/us/merchant/solutions/mastercard_securecode.html
- Cardinal Centinel: http://www.paypal-business.co.uk/3Dsecure.asp

Integration Overview

To use 3-D Secure with PayPal, you must do the following. Each item is explained in detail later in this document.

- Register your company with Cardinal Commerce and download and install the Cardinal Thin Client package.
- Insert processing for 3-D Secure into your application's debit or credit card payment flow immediately before the direct payment API request.
- Add additional fields to the direct payment API request.
- Update your website with required 3-D Secure logos, status windows, and other information for your customers.
- Test your 3-D Secure integration using Cardinal's testing facilities. PayPal's Sandbox cannot be used for testing 3-D Secure functionality.

Cardinal Commerce Registration and Installation

Before you can use Cardinal Centinel to obtain cardholder authentication:

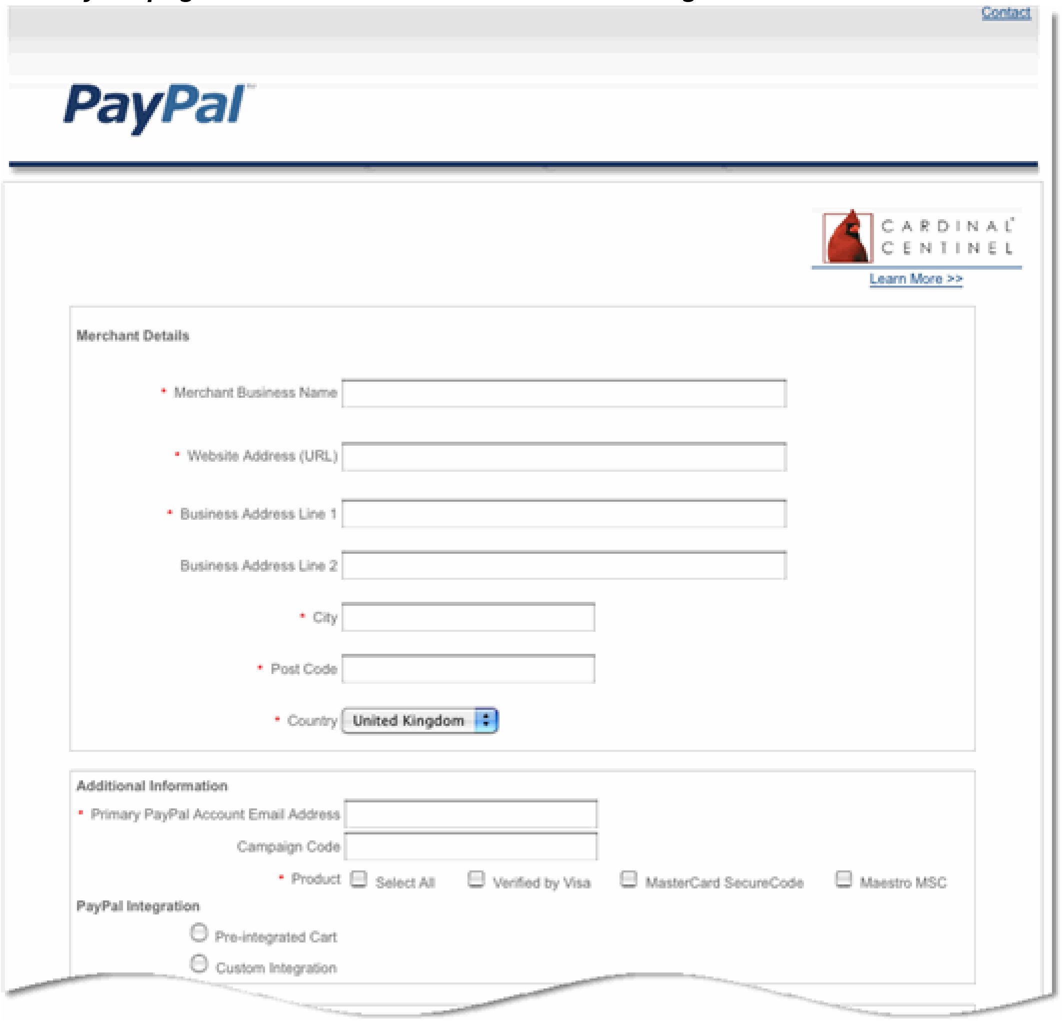
1. Register with Cardinal by filling in a simple form: http://www.paypal-business.co.uk/3Dsecure.asp.

   After you have registered, Cardinal Commerce acknowledges your 3-D Secure registration by sending you an email and welcome pack, which includes information about next steps and links for downloading their documentation and software.

2. Download and install the Cardinal Centinel Thin Client software. Refer to the Cardinal documentation for installation instructions.

NOTE: Cardinal Commerce will be available to schedule an integration meeting with you and will support you with your Cardinal Centinel Integration requirements.

PayPal page for Cardinal Commerce merchant registration


Contact

PayPal

CARDINAL CENTINEL
Learn More >>

Merchant Details

* Merchant Business Name	
* Website Address (URL)	
* Business Address Line 1	
Business Address Line 2	
* City	
* Post Code	
* Country	United Kingdom

Additional Information

* Primary PayPal Account Email Address	
Campaign Code	
* Product	☐ Select All ☐ Verified by Visa ☐ MasterCard SecureCode ☐ Maestro MSC

PayPal Integration

○ Pre-integrated Cart
○ Custom Integration

Transaction Processing

Integrating Cardinal Centinel and 3-D Secure with your PayPal transaction processing is fairly straightforward. You need to set up a web page that can handle a return call from the card's issuer, insert three additional requests into your application before the direct payment request, and add 3-D Secure payer authentication fields to the `DoDirectPayment` request.

NOTE: Refer to the Cardinal documentation for the most recent Cardinal Centinel information. Cardinal requests, responses, and processes are provided for you in this document as a convenience but might not reflect the most current Cardinal information.

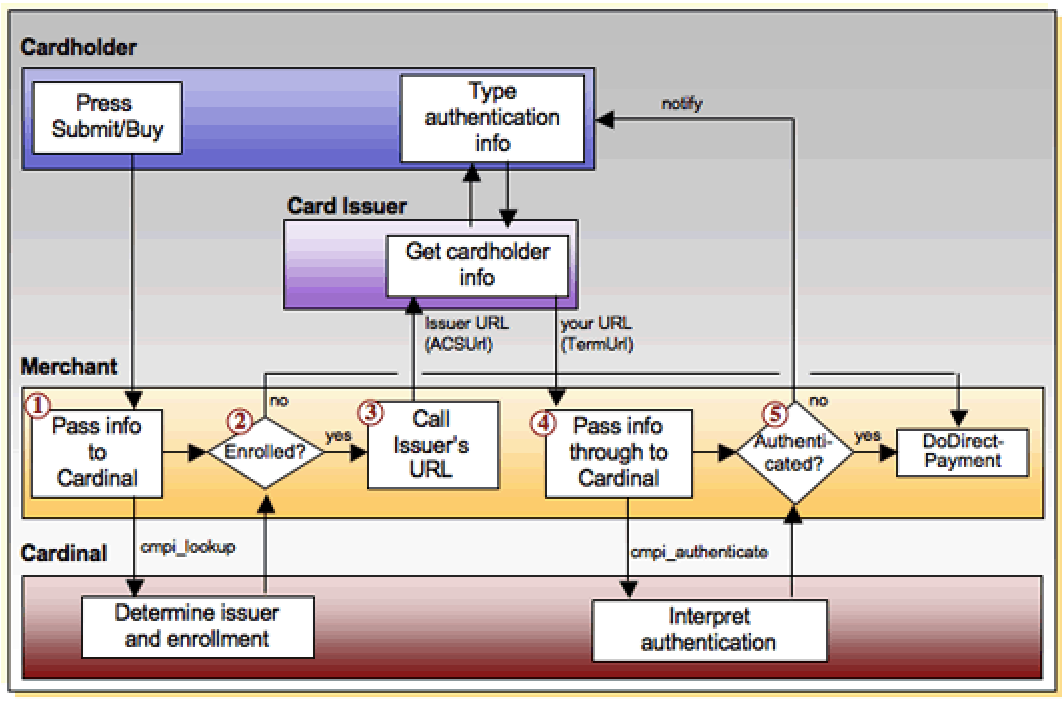
URL to Handle Issuer's Response

You must establish a page on your site whose URL can receive a form POST from the card's issuer that contains two fields, `PaRes` and `MD`. The page must then request `cmpi_authenticate` as described in the next section. Your page's URL is referred to as `TermURL`.

Transaction Flow

NOTE: The steps in this section are explained in the Cardinal *Thin Client Integration Guide Payer Authentication* document; refer there for the most current information. This summary is provided for you as a convenience.

Transaction flow with numbered steps



To create a 3-D Secure transaction using Website Payments Pro and Cardinal Centinel, do the following before executing the direct payment request:

1. Call Cardinal Centinel with the `cmpi_lookup` request, passing your merchant and transaction information.

   See "cmpi_lookup API" for the complete list of required fields.

2. The `cmpi_lookup` request responds with several fields; see "cmpi_lookup API" for details about these fields:

- Error information: Evaluate and take appropriate action if nonzero. Refer to the *Thin Client Integration Guide* for error codes, descriptions, explanations, and recommended actions.
- Cardinal transaction information that you will pass to other requests: `TransactionId`, `Enrolled`, `Payload`, and `EciFlag`.
- `ACSUrl`: If `Enrolled`=Y, this contains the URL for the card issuer's (bank's) authentication site.

Evaluate `Enrolled` and `ACSUrl`:

- If cardholder is not enrolled (`Enrolled` is N) or if the Authentication service is unavailable (`ACSUrl` is U or N), continue with Step 5 below.
- If cardholder is enrolled, continue with the next step.

3. Using an HTTP form POST, pass the cardholder to the URL (`ACSUrl`) returned by `cmpi_lookup`, which is the card issuer's authentication URL. Pass several fields to this request:

   - Transaction information from `cmpi_lookup`.
   - `MD` field optionally containing arbitrary merchant data; set to blank if not used.
   - `TermUrl`: The URL of the page you set up to handle the issuer's return call.

   See "Issuer Authentication Request" for the complete list of required fields.

   Cardholders attempt to authenticate themselves at the issuer's URL. The completion of the attempt returns a response to your application by using HTTP Form POST to call the URL you specified in `TermUrl`. The response contains the `PaRes`.

4. Call `cmpi_authenticate`, passing `PaRes` as `PAResPayLoad`. This interprets the payload to determine whether the cardholder passed the authentication process with the card's issuer.

   See "cmpi_authenticate API" for the complete list of required fields.

5. The `cmpi_authenticate` request returns several fields:

   - Error information: Evaluate and take appropriate action if nonzero. Refer to the *Thin Client Integration Guide* for error codes, descriptions, explanations, and recommended actions.
   - Authentication information in `PAResStatus`, `SignatureVerification`, `Cavv`, `EciFlag`, and `Xid`.

     See "cmpi_authenticate API" for details about the returned fields.

   Evaluate `SignatureVerification`:

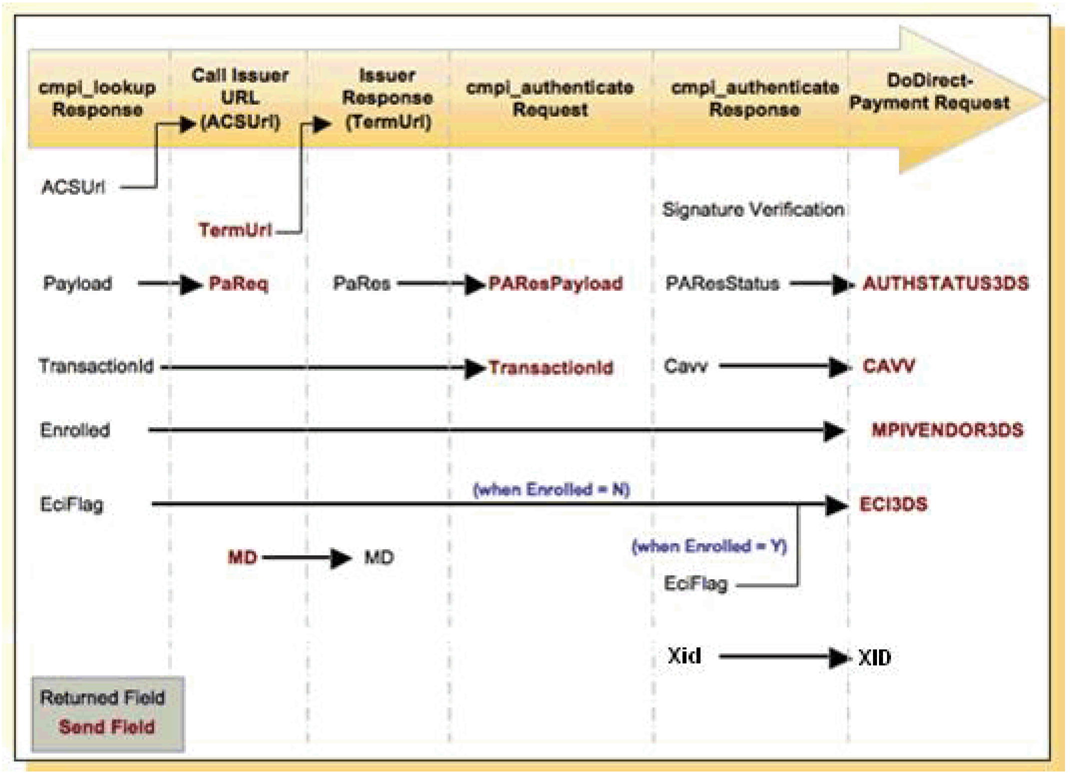
   - If the signature is not verified, return an authentication-failed message to the cardholder and stop processing the transaction.
   - If the signature is verified, continue with the next section.

3-D Secure Fields for Direct Payment Transaction Requests

If the cardholder is authenticated, or if not authenticated for valid reasons such as the authentication service being unavailable or the cardholder not being enrolled, execute the direct payment transaction request with the following additional fields:

NVP/SOAP Field	SOAP Field	Description
`VERSION`	`Version`	Set to 59.0. **You must use this version number or the request will not work.**
`AUTHSTATUS3DS`	`AuthStatus3ds`	Set this to the returned `PAResStatus` value.
`MPIVENDOR3DS`	`MpiVendor3ds`	Set to the `Enrolled` value from Step 2.
`CAVV`	`Cavv`	Set to the returned `Cavv` value.
`ECI3DS`	`Eci3ds`	Set to the returned `EciFlag` value.
`XID`	`XID`	Set to the returned `Xid` value.

Flow of field values among requests and responses. Not all fields are shown.



Example DoDirectPayment request

The following example shows the additional fields required for 3-D Secure transactions; refer to the `DoDirectPayment` documentation for all required fields.

```
METHOD=DoDirectPayment&...&ECI3DS=5&CAVV=OTJlMzViODhiOTllMjBhYmVkMGU
=&AUTHSTATUS3DS=Y&MPIVENDOR3DS=Cardinal&VERSION=59.0
```

Website Setup

Cardinal and/or 3-D Secure require that you add specific elements to your web pages. These include:

- Integrate the authentication window to allow for consistent site branding during the authentication process.
- Display the Visa and MasterCard “Learn More” logos on your home and check-out pages.
- Provide text and logo on the check-out page in which you collect payment information. Notify the cardholder that they may be required to provide their authentication password.
- Notify the cardholder of authentication results.

Refer to the Cardinal *Thin Client Integration Guide* for details.

Examples

The following examples outline the transaction process for three basic situations.

Example 1: Successful 3-D Secure Authentication

In this example, the cardholder’s issuer uses 3-D Secure and the authentication is successful:

1. Construct your message for `cmpi_lookup`.
2. Review the response from `cmpi_lookup`. Ensure that all of the following are true:
   - `ErrorNo`=0
   - `Enrolled`=Y
   - `ACSUrl` has content; for example: <ACSUrl>https://www.somewebsite.com/acs</ACSUrl>
3. Using HTTP Form POST, redirect the cardholder to the issuer’s URL that was provided in `ACSUrl`. Ensure that the `PaReq` has the data from the field `PayLoad`, from the `cmpi_lookup` response.
4. Send the response (redirect) from the issuer’s HTTP Form POST to Cardinal using the `cmpi_authenticate` message to determine how to proceed with the transaction.
5. The `cmpi_authenticate` response will specify how to proceed. To continue with authorization, the following must be true:
   - `SignatureVerification`=Y

- PAResStatus=Y, U, or A

6. Authoress as normal, with the additional fields described in 3-D Secure Fields for Direct Payment Transaction Requests.

Example 2: 3-D Secure with Unsuccessful Authentication

In this example, the cardholder's issuer uses 3-D Secure and the authentication is **not** successful:

1. Construct your message for `cmpi_lookup`.
2. Review the response from `cmpi_lookup`. Ensure that all of the following are true:
   - `ErrorNo` is 0
   - `Enrolled` is Y
   - `ACSUrl` has content; for example: <ACSUrl>https://www.somewebsite.com/acs</ACSUrl>
3. Using HTTP Form POST, redirect the cardholder to the issuer's URL that was provided in `ACSUrl`. Ensure that the `PaReq` has the data from the field `PayLoad`, from the `cmpi_lookup` response.
4. Send the response (redirect) from the issuer's HTTP Form POST to Cardinal using the `cmpi_authenticate` message to determine how to proceed with the transaction.
5. The `cmpi_authenticate` response determines how to proceed. In this example, when authentication fails, the following are true:
   - `SignatureVerification`=Y
   - `PAResStatus`=N
6. Notify the cardholder that the transaction is declined.

Example 3: Card Issuer Not Using 3-D Secure

In this example, the card's issuer does not use 3-D Secure:

1. Construct message for `cmpi_lookup`.
2. Review the response from `cmpi_lookup`. Ensure that:
   - `ErrorNo`=0
   - `Enrolled`=N or U
3. Authoress as normal, with the additional fields described in 3-D Secure Fields for Direct Payment Transaction Requests.

Example 4: Merchant Not Using 3-D Secure

In this example, the merchant does not authenticate a Maestro transaction using 3-D Secure before executing a direct-payment transaction request:

- Through 31 December, 2009, a `DoDirectPayment` request returns `ACK=SuccessWithWarning` with error code 12001; PayPal still accepts the transaction.
- Beginning on 1 January, 2010, a `DoDirectPayment` request will return `ACK=Failure` with error code 12000 and PayPal will *not* accept the transaction.

Testing

For 3-D Secure, you cannot use PayPal's sandbox for testing. You must use Cardinal's test procedures.

Refer to the Cardinal documentation.

cmpi_lookup API

The `cmpi_lookup` request is a Cardinal Centinel request. This section lists required fields and responses as a convenience for you. Refer to the Cardinal *Thin Client Integration Guide Payer Authentication* document for details and the most current information.

cmpi_lookup Request

cmpi_lookup Request Fields

Field	Description
`MsgType`	*(Required)* Must be `cmpi_lookup`.
`Version`	*(Required)* Must be 1.7.
`ProcessorId`	*(Required)* Your processor identification code, assigned by Cardinal when you register.
`MerchantId`	*(Required)* Your merchant identification code as assigned by Cardinal.
`TransactionPwd`	*(Required)* Your Cardinal password as you configured it at the Cardinal site.
`TransactionType`	*(Required)* Must be C.
`Amount`	*(Required)* The value of the transaction in cents or pence with no decimal point. For example, £100 is specified as 10000.
`CurrencyCode`	*(Required)* The 3-digit numeric ISO 4217 currency code for the sale amount. For example, GBP = 826, EUR = 978.

Field	Description
`CardNumber`	*(Required)* The debit or credit card number, up to 19 digits with no nonnumeric characters.
`CardExpMonth`	*(Required)* The card's expiration month, formatted as MM.
`CardExpYear`	*(Required)* The card's expiration year, formatted as YYYY.
`OrderNumber`	*(Required)* Your order number or transaction identifier. Limited to 50 characters.
various additional fields	*(Optional)* Cardinal allows several additional optional fields, which you can choose to use. Refer to the Cardinal documentation for details.

cmpi_lookup Response

cmpi_lookup Response Fields

Field	Description
`ErrorNo`	Error number. 0 indicates no error.
`ErrDesc`	Empty if there is no error, otherwise, describes the error.
`TransactionId`	Centinel transaction identifier. Identifies the transaction within Centinel.
`Enrolled`	Status of authentication eligibility. If not Y, then the cardholder is *not* eligible for authentication. Possible values are: ● Y: Enrolled ● N: Not enrolled ● U: Cardholder authentication unavailable.
`ACSUrl`	If Enrolled=Y, this contains the URL to which your application must next send the cardholder to complete authentication.
`Payload`	If Enrolled=Y, this contains the encoded transaction details; otherwise, this field is empty.
`EciFlag`	The Electronic Commerce Indicator (ECI). **MasterCard:** ● 01: Merchant Liability ● 02: Issuer Liability **Visa:** ● 05: Issuer Liability ● 06: Issuer Liability ● 07: Merchant Liability

Issuer Authentication Fields

The call to the card issuer's site is explained in the Cardinal *Thin Client Integration Guide Payer Authentication.* document. This section lists required fields and responses as a convenience.

Issuer Authentication Request

Specify the following fields when you call the card issuer's URL that you received from `cmpi_lookup` in `ACSUrl`.

Issuer Authentication Request Fields

Field	Description
`PaReq`	*(Required)* Content of the `Payload` field from `cmpi_lookup`.
`TermUrl`	*(Required)* Your URL for handling and processing the response from the ACSUrl.
`MD`	*(Required)* Merchant's session tracker. Set to blank if not used.

Issuer Authentication Response

When the issuer has completed its authentication processing, it calls the URL that you provided to it in `TermURL`. The issuer returns the following fields.

Issuer Authentication Response Fields

Field	Description
`PaRes`	Authentication information to pass to `cmpi_authenticate`.
`MD`	Copy of `MD` sent by merchant.

cmpi_authenticate API

The `cmpi_authenticate` request is a Cardinal Centinel request. This section lists required fields as a convenience for you. Refer to the Cardinal *Thin Client Integration Guide Payer Authentication* document for details and the most current information.

cmpi_authenticate Request

cmpi_authenticate Request Fields

Field	Description
`MsgType`	*(Required)* Must be `cmpi_authenticate`.
`Version`	*(Required)* Must be 1.7.
`ProcessorId`	*(Required)* Your Processor identification code as assigned by Cardinal.
`MerchantId`	*(Required)* Your merchant identification code as assigned by Cardinal.
`TransactionPwd`	*(Required)* Your Cardinal password as you configured it at the Cardinal site.
`TransactionType`	*(Required)* Must be C.
`TransactionId`	*(Required)* The transaction identifier returned from `cmpi_lookup`.
`PAResPayload`	*(Required)* `PaRes` provided in the package returned after the call to the card's issuer.

cmpi_authenticate Response

cmpi_authenticate Response Fields

Field	Description
`ErrorNo`	Error number. 0 indicates no error; 1140 indicates that the cardholder pressed "Back."
`ErrDesc`	Empty if there is no error, otherwise, describes the error.
`PAResStatus`	The outcome of the issuer's authentication. Possible values are: ● Y: Successful; merchant is protected. ● N: Failed; no protection. ● U: Unable to complete; no protection. ● A: Successful; merchant is protected.
`Cavv`	A random sequence of characters; this is the encoded authentication.
`SignatureVerification`	Status of authentication eligibility. If not Y, then the cardholder is *not* eligible for authentication. Possible values are: ● Y: Good ● N: Bad

Field	Description
`EciFlag`	The Electronic Commerce Indicator (ECI). **MasterCard:** ● 01: Merchant Liability ● 02: Issuer Liability **Visa:** ● 05: Issuer Liability ● 06: Issuer Liability ● 07: Merchant Liability
`Xid`	Transaction identifier from authentication.