

---

# **Amazon FPS Advanced Quick Start**

**Developer Guide**

**API Version 2008-09-17**



# Amazon FPS Advanced Quick Start: Developer Guide

Copyright © 2009 Amazon Web Services LLC or its affiliates. All rights reserved.

## Table of Contents

Welcome .....	1
What's New .....	5
Introduction to Amazon FPS Advanced Quick Start .....	6
Key Concepts .....	7
Amazon FPS Advanced Quick Start .....	7
Amazon Flexible Payments Service .....	7
Multi-Use Payment Tokens .....	9
Recurring Payment Tokens .....	11
Recipient Tokens .....	13
Other Integration Points .....	13
Cancel Payment .....	14
Refund Payment .....	14
Sender, Recipient, and Caller Actions .....	14
Request Security .....	15
Co-Branded User Interface (CBUI) .....	15
Where the CBUI Fits in the Workflow .....	16
Recipient Registration on Your Web Site .....	18
Payment Token Types .....	18
Amazon FPS API and Co-Branded Service Requests .....	20
Sandbox .....	21
Instant Payment Notification .....	21
Errors .....	22
Business Considerations .....	24
WSDLs and Schemas .....	27
Programming Guide .....	29
Important Values to Store in Your Database .....	30
Getting Authorization .....	30
Sending a Co-Branded Service Request .....	31
Recipient Registration .....	31
Making Payments .....	33
Transacting the Payment .....	34
Failed Payment Transactions .....	35
Changing the Payment Instrument .....	35
Notifications .....	36
Handling Transactions that Don't Return .....	36
Order Cancellations .....	37
Canceling a Recurring Transaction .....	37
Refunding a Recurring Transaction .....	38
Other Reversals and Issues .....	38
Testing Your Applications for Free .....	38
Testing Signatures .....	40
Working with Signatures .....	41
Verifying the ReturnURL and IPN Notifications .....	43
Soft Descriptor Customization .....	45
Setting Up Instant Payment Notification .....	47
Amazon FPS API Reference .....	53
Common Request Parameters .....	53
Common Response Elements .....	54
Actions .....	62
Cancel .....	63
CancelToken .....	65
GetTokenByCaller .....	68
GetTransactionStatus .....	71
Pay .....	74
Refund .....	81

Reserve .....	85
Settle .....	90
VerifySignature .....	93
Data Types .....	96
Enumerated Data Types .....	96
AccountBalance .....	96
ChargeFeeTo .....	97
CurrencyCode .....	97
FPSOperation .....	97
InstrumentId .....	98
InstrumentStatus .....	98
PaymentMethod .....	98
RelationType .....	99
SortOrder .....	99
TokenStatus .....	99
TokenType .....	99
TransactionalRole .....	100
TransactionStatus .....	100
Complex Data Types .....	101
Amount .....	101
AvailableBalances .....	102
DebtBalance .....	102
DescriptorPolicy .....	102
MarketplaceRefundPolicy .....	102
OutstandingDebtBalance .....	103
OutstandingPrepaidLiability .....	103
PrepaidBalance .....	103
RelatedTransaction .....	103
StatusHistory .....	104
Token .....	104
TokenUsageLimit .....	104
Transaction .....	105
TransactionDetail .....	106
TransactionPart .....	108
Co-Branded Service API Reference .....	109
Common Parameters .....	109
Recipient Token API .....	112
Recurring-Use Token API .....	114
Multi-Use Token API .....	117
Edit Token API .....	122
Code Samples .....	125
Understanding the Amazon FPS Samples .....	126
Understanding the VerifySignature Sample .....	126
Locations of the VerifySignatureSample Files in Other Libraries .....	128
Understanding the IPNAndReturnURLValidation Sample .....	133
Locations of the IPNAndReturnURLValidation Files is Other SDKs .....	134
Getting the Samples .....	135
Appendix: Verifying Responses Signed Using Signature Version 1 .....	137
Access Key Rotation Considerations with Signature Version 1 .....	137
Appendix: Moving your Application to Signature Version 2 .....	139
Differences Between Signing Versions .....	140
Signature Version 2 FAQ .....	141
Glossary .....	145
Document Conventions .....	147

---

# Welcome

---

## Topics

- [Amazon FPS Quick Starts \(p. 1\)](#)
- [Audience \(p. 2\)](#)
- [Reader Feedback \(p. 2\)](#)
- [How This Guide Is Organized \(p. 2\)](#)
- [Amazon FPS Resources \(p. 3\)](#)

This is the *Amazon FPS Advanced Quick Start Developer Guide*. This section describes who should read this guide, how the guide is organized, and other resources related to this web service.

Amazon Flexible Payment Services is occasionally referred to within this guide as "Amazon FPS", and the Amazon FPS Advanced Quick Start is often referred to as "Advanced Quick Start"; all copyrights and legal protections still apply.

## Amazon FPS Quick Starts

Amazon FPS has five parts, each providing a different slice of Amazon FPS functionality:

- **Amazon FPS Basic Quick Start**—Facilitates a one-time payment between a buyer and a developer (you) who is also the merchant for e-commerce, digital content, donations, or services.
- **Amazon FPS Marketplace Quick Start**—Facilitates a one-time payment between a buyer and a merchant, where you are a third-party developer (also known as a *caller*) who hosts the merchant's product pages and order pipeline. With this unique three-party transaction model, you can charge a fee to process transactions in which you are neither the buyer nor the merchant.
- **Amazon FPS Advanced Quick Start**—Facilitates multiple or recurring payments between a buyer and a seller for e-commerce, digital content, donations, services.
- **Amazon FPS Aggregated Payments Quick Start**—Facilitates aggregated micro-transactions into a single, larger transaction using prepaid and postpaid capabilities.
- **Amazon FPS Account Management Quick Start**—Access buyer and developer account activity programmatically. Alternatively, you can view account activity and balances on the [Amazon Payments web site](#).

You can use these parts separately or in combination. They share a common WSDL and schema. This guide covers the Advanced Quick Start.

## Audience

This guide is intended for developers who intend to enable multiple payments using Amazon FPS on their web sites or applications.

## Required Knowledge and Skills

Use of this guide assumes you are familiar with the following:

- XML (for an overview, go to the [W3 Schools XML Tutorial](#) )
- Basic understanding of web services (for an overview, go to the [W3 Schools Web Services Tutorial](#) )
- A programming language for consuming a web service and any related tools

## Business Requirements

To use Amazon FPS, you must have an Amazon FPS developer account. For information about getting the account, go to [Amazon Flexible Payments Service Getting Started Guide](#).

## Reader Feedback

The online version of this guide provides a link at the top of each page that enables you to enter feedback about this guide. We strive to make our guides as complete, error free, and easy to read as possible. You can help by giving us feedback. Thank you in advance!



## How This Guide Is Organized

This guide is organized into several major sections described in the following table.

Information	Relevant Sections
What's Changed in the documentation with this release	<a href="#">What's New (p. 5)</a>
Introduction and key concepts	<a href="#">Introduction to Amazon FPS Advanced Quick Start (p. 6)</a>
Enables merchants to register on your web site so they can receive payment for their items sold through your web site	<a href="#">Recipient Registration (p. 31)</a>
Enable buyers to authorize payments	<a href="#">Getting Authorization (p. 30)</a>

Information	Relevant Sections
Handle refunds and cancellations	<a href="#">Order Cancellations (p. 37)</a>
API reference	<ul style="list-style-type: none"><li>• <a href="#">Amazon FPS API Reference (p. 53)</a></li><li>• <a href="#">Co-Branded Service API Reference (p. 109)</a></li></ul>
Sample code for creating signatures and making a pay request	<a href="#">Code Samples (p. 125)</a>

In addition, there is a glossary and an overview of our typographical conventions. Each section is written to stand on its own, so you should be able to look up the information you need and go back to work. However, you can also read through the major sections sequentially to get in-depth knowledge about Amazon FPS Advanced Quick Start.

## Amazon FPS Resources

The following table lists related resources that you'll find useful as you work with this service.

Resource	Description
<a href="#">Amazon Flexible Payments Service Getting Started Guide</a>	Shows how to implement a simple one-time payment using Amazon FPS Basic Quick Start.
<a href="#">Amazon FPS Basic Quick Start Developer Guide</a>	Covers the one-time payment functionality of Amazon FPS.
<a href="#">Amazon FPS Marketplace Quick Start Developer Guide</a>	Covers the marketplace functionality of Amazon FPS.
<a href="#">Amazon FPS Aggregated Payments Quick Start Developer Guide</a>	Covers aggregated micro-transactions and the prepaid and postpaid functionality of Amazon FPS.
<a href="#">Amazon FPS Account Management Quick Start Developer Guide</a>	Covers the account management functionality of Amazon FPS.
<a href="#">FAQs</a>	Frequently asked questions about using Amazon FPS.
<a href="#">Release Notes</a>	The Release Notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
<a href="#">FPS Developer Resource Center</a>	A starting point specifically for FPS, to find documentation, code samples, release notes, and other information to help you build innovative applications.
<a href="#">AWS Developer Resource Center</a>	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
<a href="#">Discussion Forums</a>	A community-based forum for developers to discuss technical questions related to Amazon FPS.
<a href="#">AWS Support Center</a>	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and Premium Support (if you are subscribed to this program).

**Amazon FPS Advanced Quick Start Developer Guide**  
**Amazon FPS Resources**

---

<b>Resource</b>	<b>Description</b>
<a href="#">Product information about Amazon FPS</a>	The primary web page for information about Amazon FPS.
<a href="#">Contact Us</a>	A central contact point for inquiries concerning AWS billing, account, events, abuse, etc.
<a href="#">Conditions of Use</a>	Detailed information about the copyright and trademark usage at Amazon.com and other topics.



# What's New

This What's New is associated with the 2008-09-17 version of the *Amazon FPS Advanced Quick Start*. This guide was last updated on 2009-11-06.

The following table describes the important changes since the last release of this guide.

Change	Description	Release Date
New Feature	<p>support for signature version 2, which will completely replace signature version 1 on 01 November, 2010. The enhanced security features include:</p> <ul style="list-style-type: none"> <li>• a more secure way of calculating signatures for <i>inbound requests</i> and <i>outbound notifications</i>. For more information, see <a href="#">Working with Signatures (p. 41)</a>.</li> <li>• support for <i>SHA256</i> signing algorithm</li> <li>• the new <code>VerifySignature</code> FPS Action for server-side testing of return URL responses and IPN notifications. For more information, see <a href="#">VerifySignature (p. 93)</a>.</li> <li>• support for <i>PKI</i> based authentication for client-side testing of return URL responses and IPN notification. For more information, see <a href="#">Client-side Signature Validation (p. 44)</a>.</li> </ul>	2009-11-03
Enhancement	The Access Keys page has been renamed the Security Credentials page, located at <a href="https://aws-portal.amazon.com/security-credentials">https://aws-portal.amazon.com/security-credentials</a> .	2009-09-09
Enhancement	The <code>GetTransaction</code> action has been enhanced to return information sent asynchronously by the backend processor (through both the <code>returnUrl</code> and IPN notification) for <code>Pay</code> , <code>Reserve</code> , <code>Settle</code> , and <code>Retry</code> API calls. For more information, see .	2009-09-25

# Introduction to Amazon FPS Advanced Quick Start

---

This introduction to Amazon FPS Advanced Quick Start provides a detailed summary of this web service. After reading this section, you should have a good idea of what it offers and how it can fit in with your business.

## Overview

This overview describes the business model and major features of Amazon FPS Advanced Quick Start.

### Business Model

Amazon FPS Advanced Quick Start enables buyers to authorize a payment token that can be used more than one time. For more information about payment tokens, see [Payment Token Types \(p. 18\)](#).

These payment tokens, which are set up with a certain amount of money authorized, might be recurring payment tokens, which are charged a fixed price at regular intervals. Or they could be multi-use tokens, which are charged according to the token's constraints. For example, the token could be charged a certain number of times per day, or for a certain number of users, or until the authorization runs out of money.

These token types are highly configurable. For a recurring payment token, you can configure how often and how much is paid, and when the token expires. For a multiple payment token, you can configure its expiration, the merchants it can be used to pay, the maximum and minimum amount it can be charged at any one time, how many times it can be charged, the lifetime maximum amount it can be charged, and so on.

### Features

Amazon FPS Advanced Quick Start provides the following major features:

- **Payment authorization**—Buyers must authorize payments for you to be able to charge them. You can get this authorization by redirecting them to the Amazon Payments website. Buyers log in to Amazon Payments and agree to make the payment using a specified payment method.
- **Payments**—Transfer money from the buyer's account to the seller's.
- **Recurring-use payment tokens**—A buyer authorizes a recurring payment token once, and thereafter it is charged a specific amount at regular intervals until the token expires.
- **Multi-use payment tokens**—A buyer authorizes a multi-use payment token once but it can be used repeatedly according to its constraints, such as the maximum number of times it can be used within a given period of time.
- **Notifications**—Get notified automatically when transactions succeed or fail.
- **Refunds**—Refund the money from a successfully completed transaction.
- **Cancellation of payment tokens**—Cancel any of your payment tokens at any time.

## Key Concepts

### Topics

- [Amazon FPS Advanced Quick Start \(p. 7\)](#)
- [Multi-Use Payment Tokens \(p. 9\)](#)
- [Recurring Payment Tokens \(p. 11\)](#)
- [Recipient Tokens \(p. 13\)](#)
- [Other Integration Points \(p. 13\)](#)
- [Sender, Recipient, and Caller Actions \(p. 14\)](#)
- [Request Security \(p. 15\)](#)
- [Co-Branded User Interface \(CBUI\) \(p. 15\)](#)
- [Sandbox \(p. 21\)](#)
- [Instant Payment Notification \(p. 21\)](#)
- [Errors \(p. 22\)](#)
- [Business Considerations \(p. 24\)](#)
- [WSDLs and Schemas \(p. 27\)](#)

This section describes the concepts and terminology you need to understand to use Amazon FPS Advanced Quick Start effectively.

## Amazon FPS Advanced Quick Start

Amazon FPS Advanced Quick Start enables you to create multi-use payment tokens with only a single authorization, and to create innovative payment solutions that allow payments between two applications.

## Amazon Flexible Payments Service

The Quick Start implementation covered in this guide is one of five different Quick Start implementations that make up the Amazon Flexible Payments Service. Amazon FPS is the first payments service designed from the ground up specifically for developers. This set of web service APIs differs from other Amazon Payments products, such as Amazon Simple Pay and Checkout by Amazon, in that it allows the development of highly customized payment solutions for a variety of

businesses. Amazon FPS is built on top of Amazon's reliable and scalable payments infrastructure and provides developers with a convenient way to charge the tens of millions of Amazon customers. Amazon customers can pay using the same login credentials, shipping address and payment information they already have on file with Amazon.

For buyers, the advantage of using Amazon FPS payment instruments in online purchases includes the following:

- **Convenience**—Consumers can use their Amazon.com account to complete payments on a web site without having to re-enter their shipping address or payment information.
- **Trusted payment experience**—The secure and trusted payment experience consumers enjoy on Amazon.com is available for your web site.
- **Purchase protection for buyers**—Consumers can feel more confident purchasing, knowing that they have the same protection under the Amazon A-to-z Guarantee that they have when they shop on Amazon.com.

For sellers, the advantage of using Amazon FPS includes the following:

- **Flexibility**—Amazon FPS offers immense flexibility by allowing you to define terms and conditions specific to each transaction. It also gives you control over when the payment transaction is executed.
- **Access to Amazon customers**—Amazon FPS enables tens of millions of existing Amazon customers to transact online, simply using the same accounts and payment methods that they use for purchases on Amazon.com.
- **Increased customer base**—Amazon's trusted payment experience, A-to-z Guarantee, and the ease with which tens of millions of Amazon customers can pay on a web site will help increase the total number of Amazon customers.
- **Lower cost with Amazon's proven fraud detection**—Amazon FPS leverages Amazon's proven fraud detection capabilities, chargeback controls, and risk management processes to reduce bad debt.
- **Reliable and secure payments platform**—Amazon has spent over a decade developing, testing, and operating a reliable, scalable and secure payments infrastructure to support millions of daily transactions.

Amazon FPS exposes this robust infrastructure to you and your customers.

Amazon FPS has five Quick Start implementations, each providing a different slice of Amazon FPS functionality:

Name	Description
<b>Basic</b>	Facilitates one time payment between a buyer and a developer who is also the merchant for e-commerce, digital content, donations, services.
<b>Marketplace</b>	Facilitates one time payment between buyer and merchant where you are a third party developer, a caller, who hosts the merchant's products and <a href="#">order pipeline</a> . With a unique three-party transaction model, payments can be processed in which you are neither the buyer nor the seller. You can charge a fee for such transactions.
<b>Advanced</b>	Facilitates multiple or recurring payments.
<b>Aggregated Payments</b>	Facilitates aggregated micro-transactions into a single, larger transaction using prepaid and postpaid capabilities.
<b>Account Management</b>	Accesses buyer and developer account activity programmatically. Alternatively, view account activity and balances can be viewed on the <a href="#">Amazon Payments web site</a> .

You can use these parts separately or in combination. They share a common WSDL and schema.

## Multi-Use Payment Tokens

The multi-use payment token is a usage-based payment instrument that you can constrain in a few ways or in many ways. For example, you can constrain the allowed charge amount per time period, (such as a day, a week, a month, or the token's lifetime). You can also constrain the token to pay at least a minimum amount per transaction, or to be used only a specific number of times in any time period. Or, you can use multi-use payment tokens to pay more than one recipient.

You can use the multi-use payment token for open-ended payments where the total is not known beforehand. You might own a music download company that charges a set fee per month for membership and a fee for each download. Before the subscriber downloads each song, you add the download fee to the monthly membership fee and arrive at the current monthly bill. If the total is within the prescribed boundaries of the multi-use payment token, you can allow the music download and charge the subscriber. If the total exceeds the maximum authorized dollar amount associated with the payment token, you display a message and refuse to download unless the subscriber authorizes additional spending.

You, as the caller, can use this token to charge the sender multiple times to pay one or multiple recipients. You cannot use a multi-use token to charge the sender for an unlimited amount or for unlimited amount of time.



### Note

If you charge your customers at a regular interval for a fixed amount, you should use a recurring token instead. For more information, see [Recurring Payment Tokens \(p. 11\)](#).

## Usage Restrictions

A multi-use token provides the flexibility of usage based on restrictions or limitations. You can specify two types of restrictions for a multi-use token:

- **Amount Limit**—Limiting the maximum amount  
This specifies the maximum amount that can be charged using the token within a specified period of time or for the lifetime of the token. For example, the token can be used for a maximum amount of \$30 in a month starting from August 10, 2008 or for a maximum amount of \$30.
- **Usage Limit**—Limiting the number of uses  
This specifies the maximum number of times you can charge the token within a specified period of time or for the lifetime of the token. For example, the token can be used a maximum of 10 times or for 10 times per day starting from August 10, 2008.

A maximum of three usage restrictions can be specified for a multi-use token. For example, a combination of usage restrictions mentioned in the preceding list restricts a token to be used 10 times a day for a maximum of \$30 in a month.



### Important

Amazon FPS requires you to always set the maximum amount limit.

In addition to the usage restrictions, you can also set one of the following restrictions on the transaction amount. These apply individually to each transaction and not across multiple transactions:

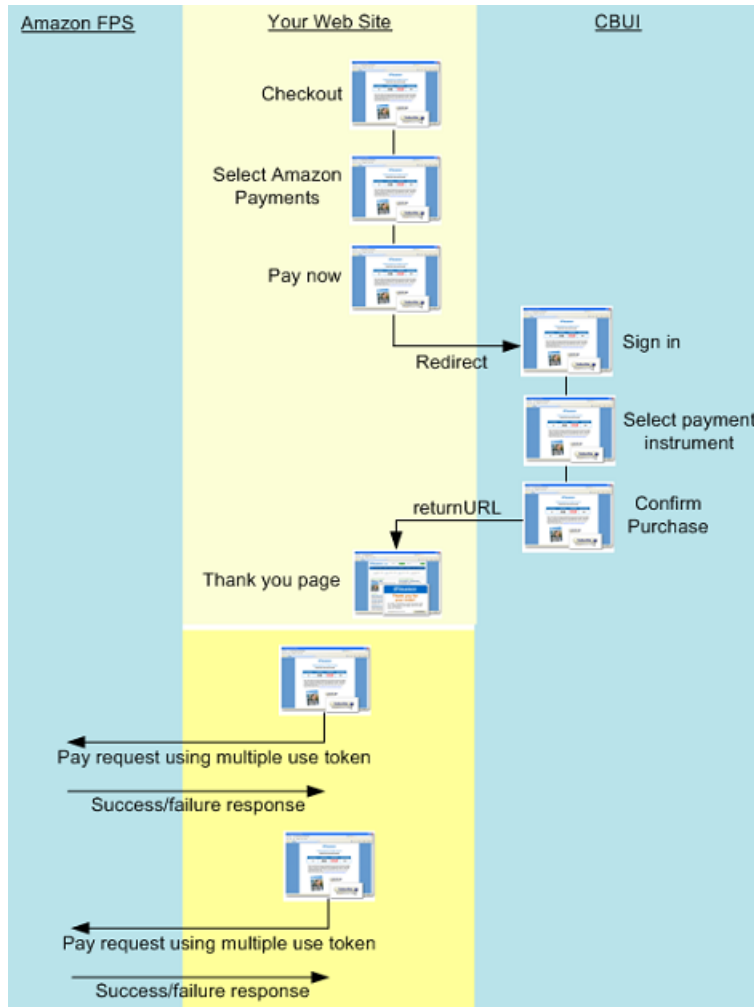
- Maximum transaction amount for each transaction
- Minimum transaction amount for each transaction

- Exact transaction amount for each transaction

A multi-use token can be used to pay multiple recipients in a marketplace application. To enable this, the calling application should specify the list of recipients that will receive payments. Once the list of recipients are specified in the token, it cannot be modified.

## Buyer's Experience of a Multi-Use Payment Token

The following figure shows the typical buyer experience of authorizing the multiple payment token and then, later, purchasing items within the limits of that payment token at different times.



### Authorizing and Using a Multi-Use Payment Token

1	Your web site takes the buyer through the checkout process.
2	If the buyer chooses Amazon Payments, your <b>Pay Now</b> button code constructs and sends a Co-Branded service request that includes the multi-use payment token parameters. The <b>Pay Now</b> button code also redirects the buyer to the CBUI web pages that Amazon hosts.
3	The buyer selects a payment instrument and authorizes the purchase.

4	The CBUI redirects the buyer to the URL specified in the <i>returnURL</i> from the Co-Branded service request. In addition to that URL, the URI contains additional parameters such as the ID for the multi-use token just created (which you need later), the status of the payment authorization, and the identification of the buyer.
5	When the status changes to <i>Success</i> you can use the payment token.
6	At some point, you use the payment token for a purchase without getting additional authorization from the buyer. To facilitate the purchase, you send a <i>Pay</i> request with the multi-use token ID you received earlier.
6	Later, you send additional <i>Pay</i> requests to make additional purchases, as necessary.

## Recurring Payment Tokens

Recurring-use payment tokens are a subset of multi-use payment tokens, but with the following restriction types:

- Pre-determined fixed amount (specified at the time of token creation)
- Regular interval

An example would be a subscription for a magazine that charges users \$9.99 every month for one year starting on March 10, 2008.

The recurring-use payment token can be charged on a recurring basis, but the buyer only authorizes the payment once. You can customize the token by configuring a variety of parameters. You can set the amount of the recurring payment, the frequency of payment, the starting date of the payments, and the expiration of payments in each payment token.

The advantage of FPS is that you have full control over the charges. For example, you might already be charging a credit card for subscriptions, and you might want to manage an additional payment method, without changing your business processes and control when the payment actually happens.



### Important

You must make a *Pay* request each time you want to charge a recurring payment token. The constraints associated with the recurring payment token restrict its use only; they do not set up automatic payments.

You might set up recurring payment tokens to pay for such things as membership fees, online newsletters, donations, and loan payments.

## Buyer's Experience of a Recurring Payment Token

The following table shows the sequence of events a buyer goes through while purchasing a music download subscription.

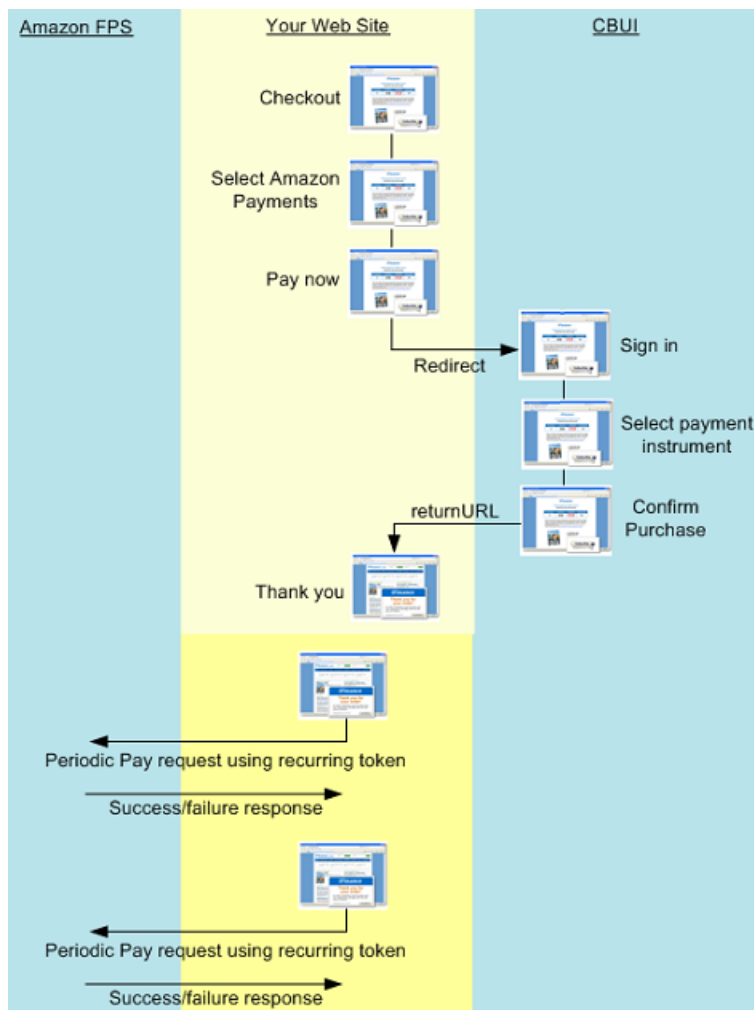
### Purchasing a Music Download Subscription

1	The buyer goes to the music subscription web site and clicks a <b>Subscribe Now</b> button (or equivalent wording) to purchase the music subscription.
2	The buyer is redirected to the Amazon-hosted Co-Branded User Interface (CBUI) pages where he or she logs in, selects the payment instrument, reviews the costs and payment structure, and authorizes the purchase of the subscription.

3	The buyer is redirected to the URL you specify in the Co-Branded service request, which is typically a thank you page and an invitation to keep shopping.
4	After the developer has issued the Pay request, the buyer notices that their payment instrument is charged on the start date of the subscription and at the prescribed intervals thereafter.

## Recurring Payments

The first point where you must integrate Amazon FPS is the point where buyer decides to purchase, as shown in the following diagram. For more information, see [Getting Authorization \(p. 30\)](#).



### Authorizing and Using a Recurring Payment Token

1	Your web site takes the buyer through the checkout process.
2	If the buyer chooses Amazon Payments, your <b>Pay Now</b> button code constructs and sends a Co-Branded service request that includes the recurring payment token parameters. The <b>Pay Now</b> button code redirects the buyer to the CBUI web pages that Amazon hosts.



3	The buyer selects a payment instrument and authorizes the purchase.
4	The CBUI redirects the buyer to the URL specified in the <code>returnURL</code> from the Co-Branded service request. In addition to that URL, the URI contains additional parameters such as the token ID for the token just created (which you need in the next step), the status of the payment authorization, and the identification of the buyer.
5	When the status changes to <code>Success</code> , your code constructs and sends Amazon FPS a <code>Pay</code> request to initiate the purchase. The request includes the token ID received in the previous step. You should save the transaction ID that is returned in case of a refund or chargeback.
6	Later, you send additional <code>Pay</code> requests at regular intervals based on the parameters in the Co-Branded service request.

## Recipient Tokens

Amazon FPS Advanced Quick Start includes the option of using [marketplace](#) functionality with recurring-use and multi-use payment tokens. In the marketplace environment, you function as a third-party caller who makes Amazon FPS web service calls to transfer money between a sender and a recipient. To implement marketplace functionality, you must register the recipient (also called the seller or the merchant) on your web site. Your registration process creates a *recipient token* that you later use in a `Pay` request to transfer money from the buyer to the seller.

The following diagram describes the marketplace workflow for the merchant.



A recipient token contains the following information:

- Accepted payment method
- Agreement to pay fees (if you don't pay the associated fees)
- Validity of the token
- Authorization for you as the caller to transfer money

## Other Integration Points

### Topics

- [Cancel Payment \(p. 14\)](#)
- [Refund Payment \(p. 14\)](#)

The preceding sections explain how recurring and multi-use payment tokens fit into the workflow of your web site. The following sections show the other places where Amazon FPS Advanced Quick Start fits into the workflow of your web site.

## Cancel Payment

Buyers can cancel a recurring or multi-use payment token, as shown in the following diagram.



### Canceling Interaction

1	The buyer clicks the equivalent of a <b>Cancel</b> button hosted on your web site.
2	Your <b>Cancel</b> button code constructs and sends a <code>CancelToken</code> request, which includes the <code>tokenID</code> returned in the Co-Branded service response.
3	Amazon FPS processes the request and returns an XML notification of the request's success or failure.

## Refund Payment

Buyers cannot request a refund through the buyer's account page on Amazon Payments. This must be handled on your Web site and you must send a request to Amazon Payments.

There are two ways you can refund a payment to your customers. First, you can visit your account on the Amazon Payment web site to refund a payment. Or, you can issue refunds programmatically when your customers want a refund.

## Sender, Recipient, and Caller Actions

Participants involved in an Amazon FPS transaction perform one or more of the following actions:

- **Send money**  
The buyer, known as the *sender* in an Amazon FPS transaction, makes the payment for purchasing goods or services. The sender can send money using an Amazon Payments Personal account, Amazon Payments Business account, or Amazon FPS developer account.
- **Receive money**  
The merchant (or seller), also known as the *recipient* in an Amazon FPS transaction, receives payment for the goods or services sold to the sender. A recipient can receive money using an

Amazon Payments Personal account, Amazon Payments Business account, or Amazon FPS developer account.

- Make Amazon web service calls to enable money transfer

The developer, also known as the *caller* in an Amazon FPS transaction, can transfer money between a sender and a recipient in a transaction. A caller can also perform the role of a sender or a recipient. A caller must have an Amazon FPS developer account to make web service API calls. For more information about registering for an Amazon FPS account, go to the [Amazon Flexible Payments Service Getting Started Guide](#).



### Important

Amazon FPS *does not* allow a participant to play all three roles in a single transaction.

## Request Security

Amazon FPS applications enable payments between buyers and sellers. Web service requests are sent over the Internet using SSL (HTTPS).

HTTPS does not establish the identity of the requester. To establish the identity of the requester, Amazon FPS uses a *signature*.

A signature is an encrypted value that you generate and include as a parameter value in every request using the *signature* parameter as in the following example.

```
Signature=K2ryWe7s/0AHI0/PbuAveuUPksTefhmNCzDTold2VYA=
```

With signature version 2, you have the option of using either [SHA256](#) or [SHA1](#) for signature authentication in [inbound requests](#). For [outbound notifications](#), the RSA-SHA1 algorithm is supported.



### Important

The previous method for signing will expire on 01 November, 2010. At that time, any signing you do with your access keys must be done using the new method.

Signing is required for all FPS API requests (except for [VerifySignature](#) (p. 93)), and optional but recommended for Co-Branded service requests. If you do not sign a Co-Branded service request, you must manually determine whether the request was tampered with. For detailed information about generating a signature, see [Working with Signatures](#) (p. 41).

## Co-Branded User Interface (CBUI)

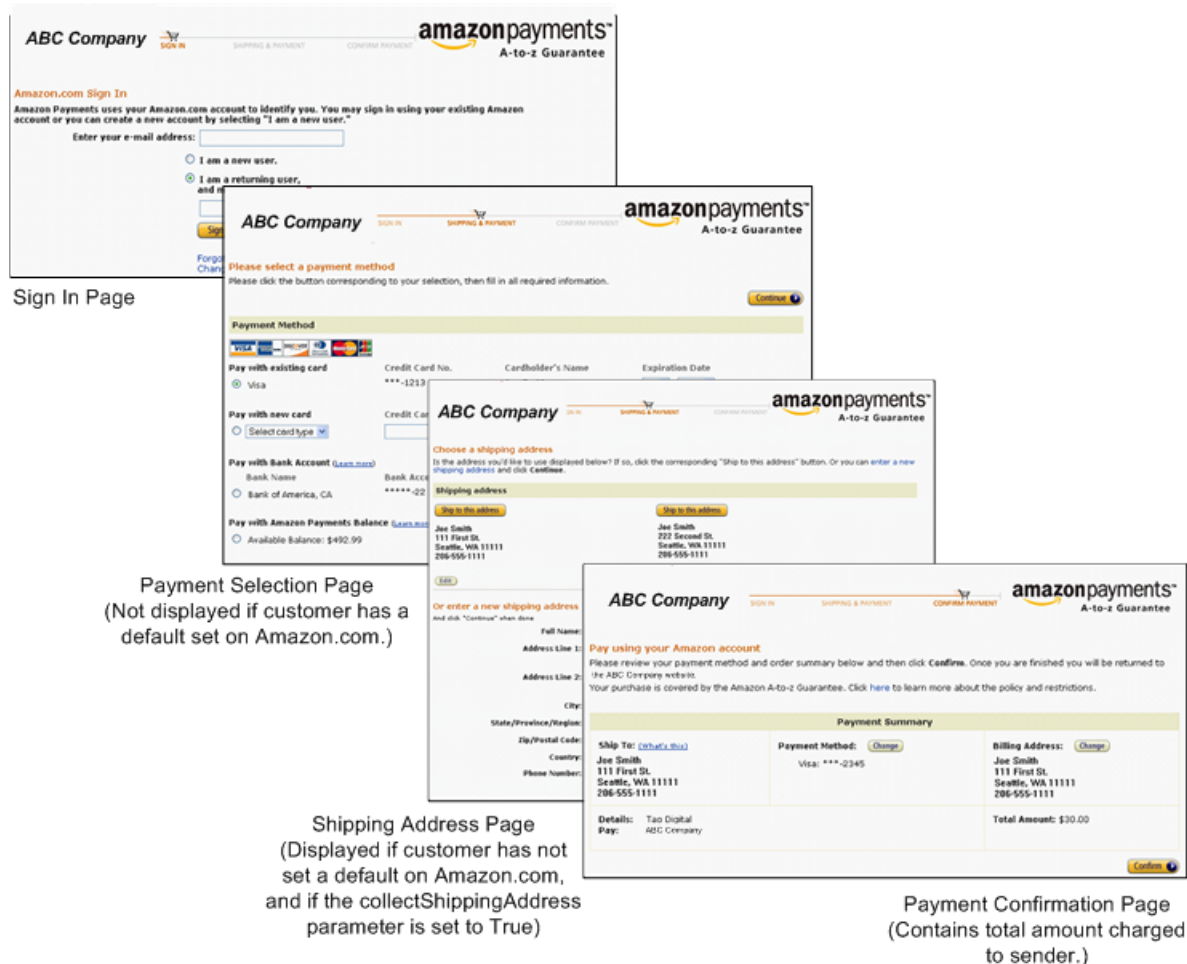
### Topics

- [Where the CBUI Fits in the Workflow](#) (p. 16)
- [Recipient Registration on Your Web Site](#) (p. 18)
- [Payment Token Types](#) (p. 18)
- [Amazon FPS API and Co-Branded Service Requests](#) (p. 20)

When someone is ready to authorize a purchase, he or she clicks the equivalent of a *Pay Now* button powered by Amazon FPS to authorize a payment. The implementation of this button is to redirect the buyer from your web site to the Co-Branded User Interface (CBUI). You cannot issue an Amazon FPS `Pay` request until a buyer has successfully completed the CBUI web pages thereby authorizing the purchase.

## Amazon FPS Advanced Quick Start Developer Guide Co-Branded User Interface (CBUI)

The CBUI is a series of web pages, as shown in the following figure.



For the buyer, the CBUI is a series of web pages they use to authorize the payment. The CBUI web pages ask the buyer to sign in, specify a personal payment instrument, like a credit card, and authorize the purchase. If you have purchased something on Amazon.com, you're familiar with the final approval in the checkout process where you commit to spending your money.

For the merchant, the CBUI is a series of web pages in which the merchant registers with a caller for a marketplace storefront on the caller's web site. Merchant registration is only required in marketplace selling environments. You use the recipient token ID returned from that request to pay merchants in the purchase transaction.

The CBUI enables you to include your company's branding on the CBUI payment authorization web pages. This makes for a better buying experience. Clicking a *Pay Now* button powered by Amazon FPS redirects the buyer away from your web site to Amazon's. By including your branding on Amazon's CBUI web pages, the buyer doesn't feel as if they've completely left the your web site to authorize a payment. The CBUI provides continuity between the checkout and payment authorization experience.

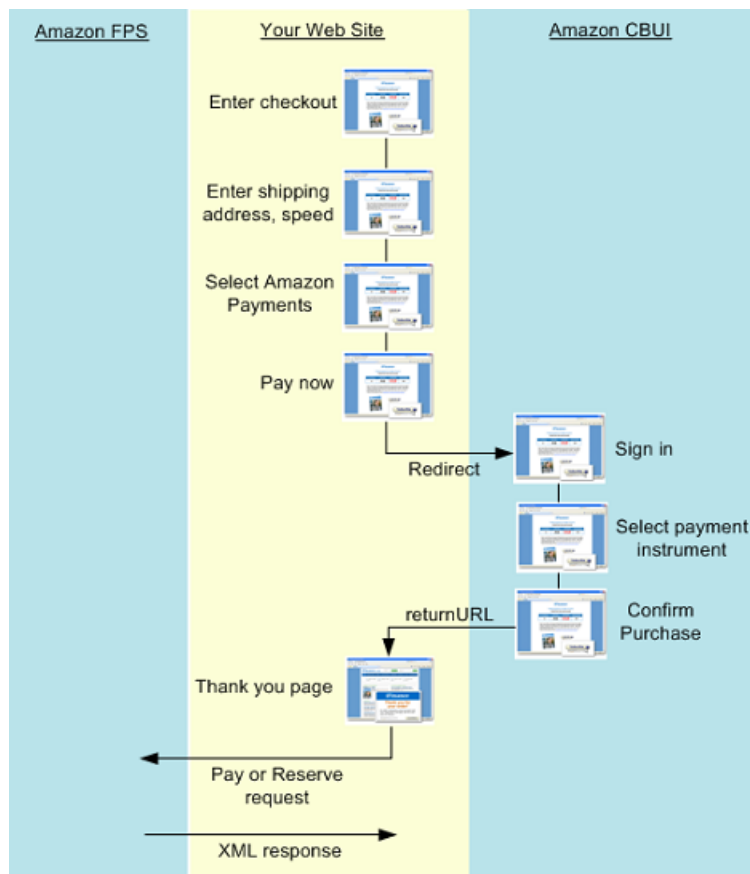
For merchant registration, co-branding provides a similar, improved customer experience.

## Where the CBUI Fits in the Workflow

The following figure shows that you redirect buyers to the CBUI web pages when they are ready to purchase the items they selected on your web site. Your web site code constructs a Co-Branded

## Amazon FPS Advanced Quick Start Developer Guide Co-Branded User Interface (CBUI)

service request that identifies the buyer, and sends it when you redirect the buyer to the CBUI web pages.



The following table describes the CBUI web pages, the authorization process, and the subsequent Amazon FPS request you make after receiving notification of the authorization.

### Authorization and Transaction Process

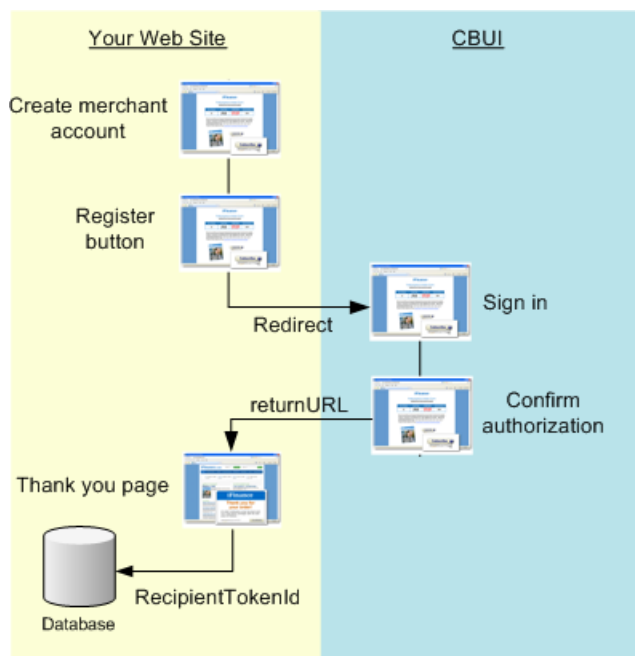
1	Senders sign into their Amazon Payments account by entering their e-mail and password.
2	They choose a payment instrument, such as a credit card, bank account, or Amazon Payments balance transfer, to make the purchase.
3	They review the transaction details and click <b>Confirm</b> to authorize the payment using the specified payment instrument.
4	The Co-Branded service creates a payment token and redirects the sender to the URL you specify in your Co-Branded service request using the <i>returnURL</i> parameter. Typically, it is a <i>Thank you page</i> on your web site in which you invite senders to keep shopping perhaps by showing them similar items to what they purchased. The URI contains not only the endpoint that you specified in <i>returnURL</i> , but also a reference to the payment token, such as a <i>tokenId</i> , and the status of the authorization.
5	Upon receiving the URI from the Co-Branded service, if the status of the authorization is successful, you must send Amazon FPS a <i>Pay</i> (or <i>Reserve</i> ) request to actually transfer money from the buyer to the merchant. This request requires, as a parameter, the <i>tokenId</i> returned by the Co-Branded service in the URI.

## Recipient Registration on Your Web Site

The first step in the workflow is recipient registration on your web site. Recipients must register with you so that:

- They can accept your business terms, in particular, the marketplace fee you will charge them
- They can upload their item information to your web site
- You can get the *RecipientTokenID* which you need to facilitate the payment to the recipient

The following figure shows the process for recipient registration.



### Process for Recipient Registration

1	Implement a recipient account system on your web site. For example, enable each recipient to sign in to their account with a sign-in name and password.
2	On the CBUI pages, the recipient selects a payment instrument to use to receive payments.
3	The Co-Branded service redirects the recipient back to your web site with information that you should store in your database, such as the <i>RecipientTokenID</i> .
4	After confirming his or her choices, the recipient is redirected back to your web site with information, such as the <i>RecipientTokenId</i> that you should store in your database.

## Payment Token Types

When someone successfully completes the CBUI web pages, the Co-Branded service creates a payment token, which is stored on Amazon servers. A payment token contains purchase information, including the amount of the purchase, the buyer, and the authorization to use the token as a means of making a purchase. Every Amazon FPS payment transaction requires a payment token.

There are a number of different kinds of payment tokens and each one behaves differently. Amazon FPS provides the following token types. Each Amazon FPS Quick Start implementation provides a different subset of them all.

- **Single-use**—Authorized to make a single purchase of a specified amount where the money is sent from the buyer to you  
Available for: Basic Quick Start
- **Recurring-use**—Authorized to make payments at regular intervals for such things as subscription purchases  
This token can have usage limitations, for example, an expiration date. The payment can be made to you or a third party merchant. In this case, you broker the deal and collect a marketplace fee for doing so. This scenario, in which there are three parties involved, buyer, merchant, and you, is called a marketplace scenario.  
This token can be used in a marketplace scenario.  
Available for: Advanced Quick Start
- **Multi-use**—Authorized to be used one or more times within its specified limitations, for example, the total amount it can be used for, how long it can be used, or how little or how much any single payment can be  
This token can be used in a marketplace scenario.  
Available: Advanced Quick Start
- **Prepaid**—Authorized to be used one or more times within its specified limitations, for example, the total amount it can be used for, how long it can be used, or how little or how much any single payment can be  
The prepaid token is funded before it is used.  
Available for: Aggregated Payments Quick Start
- **Postpaid**—Authorized to be used one or more times within its specified limitations, for example, the total amount it can be used for, how long it can be used, or how little or how much any single payment can be  
This token is like a credit card by which you agree to pay for purchases you make at some time after the actual sale.  
Available for: Aggregated Payments Quick Start
- **Editing**—Authorizes the change of an existing token  
The multiuse, recurring, and settlement tokens can have the ID edited. This feature is used to change information in an existing token, for example, the credit card number on a recurring token. If a credit card expires or is replaced, you can use the edit token to modify the recurring token information without having to force the buyer to cancel and re-purchase.  
Available for: Advanced Quick Start, Aggregated Payments Quick Start

Before you can initiate any Amazon FPS payment transaction, such as a `Pay` request, you must create at least one of these tokens.

## Sender and Recipient Token Associations

Senders (buyers) and, in Amazon FPS Quick Start implementations that support the marketplace scenario, recipients (merchants) can go through the CBUI to create tokens. Each one does so for a different purpose. The sender uses a *Pay Now* button to go through the CBUI to authorize a purchase. The recipient uses a *Register Now* button to authorize the payment of marketplace fees to you for hosting his or her e-commerce store. In both cases, it is your web site that implements the button that redirect the person to the CBUI.

All of the token types can be associated with a sender, that is, a buyer who is authorizing a purchase. So, there can be a sender single use payment token, sender recurring use payment token, sender postpaid payment token, and so forth. This guide sometimes shortens these names to "sender token." The value returned in `TokenId` from the CBUI is used as the value for `SenderTokenId` in subsequent Amazon FPS requests.

The token types that can be used in the marketplace scenario can also be associated with a recipient. In this scenario, you host the e-commerce store of a merchant, called a recipient (the person who receives the money). You charge the recipient a fee (called a marketplace fee) for hosting their e-commerce store and brokering the money transactions. On your web site, you implement a button that makes the recipient go through the CBUI and authorize the payment of marketplace fees for your service. The value returned in *TokenId* from the CBUI is used as the value for *RecipientTokenId* in subsequent Amazon FPS requests. The following token types can be associated with a recipient: single-use, multiple-use, and recurring-use.

## Token Creation

The Co-Branded service creates a token when a buyer successfully completes the CBUI web pages and thereby authorizes a purchase, or when a merchant authorizes the payment of marketplace fees to you, as shown in the following figure. The CBUI returns to your web site references to the created tokens in the *tokenID* parameter. This value is either used as a *SenderTokenID* or *RecipientTokenID*, depending on the implementation, in subsequent Amazon FPS requests.



The token type you create depends on the parameters included in the Co-Branded service request. This guide presents the API for each token type available in this Amazon FPS Quick Start. For more information about sending a Co-Branded service request, which can result in token creation, see [Getting Authorization](#) (p. 30).

## Amazon FPS API and Co-Branded Service Requests

Amazon FPS has two production endpoints where you send requests. One is for requests involving the Amazon FPS API. These requests implement all of the financial functionality included in Amazon FPS, such as *Pay* and *Refund*. The other endpoint is for Co-Branded service requests that redirect a buyer to a series of Amazon-hosted web pages where the buyer authorizes a payment.

Amazon FPS API and Co-Branded service requests differ in the following ways:



- API requests carry out actions using the Amazon FPS web service. Co-branded service requests make the buyer interact with Amazon-hosted interface in which the buyer authorizes payments, such as when he or she authorizes the use of his or her credit card to complete a purchase.
  - The response to an Amazon FPS request is an XML document. The response to a co-branded service request is a URI sent to a URL specified in the request.
  - The requests have different endpoints, as follows.
    - **Amazon FPS API**— <https://fps.amazonaws.com>
    - **Amazon Co-Branded service API**— <https://authorize.payments.amazon.com/cobranded-ui/actions/start>
- For more information about Co-Branded service requests, see [Getting Authorization \(p. 30\)](#).

You must make Co-Branded service requests before API requests because the Co-Branded service creates the payment token that you must use in API requests. The Co-Branded service returns pointers to those tokens in the form of token IDs.

## Sandbox

Amazon FPS provides an environment called the sandbox for testing your applications. In the sandbox you can try out your requests without incurring charges or making purchases. We recommend that you test all of your requests in the sandbox before exposing them on your web site.

The sandbox has two endpoints: one for the Amazon FPS API, and one for the Co-Branded service API.

- **Amazon FPS API**—<https://fps.sandbox.amazonaws.com>
- **Co-Branded service**—<https://authorize.payments-sandbox.amazon.com/cobranded-ui/actions/start>

For information about getting a sandbox account, go to the [Amazon Flexible Payments Service Getting Started Guide](#).

## Instant Payment Notification

Instant Payment Notification (IPN) is a notification mechanism that uses HTTP POST to send you immediate updates on transactions. IPN saves you the trouble of polling Amazon FPS for transaction results that complete asynchronously.

Amazon FPS sends you an IPN whenever a transaction completes, as in the following cases:

- A payment or reserve succeeds
- A payment or reserve fails
- A payment or reserve goes into a pending state
- A reserved payment is settled successfully
- A reserved payment is not settled successfully
- A refund succeeds
- A refund fails
- A refund goes into a pending state
- A payment is canceled
- A reserve is canceled
- A token is canceled successfully
- A refund succeeds
- A refund fails

- A token is canceled successfully



### Note

IPN must be configured in order to operate. If IPN is not configured, email is the only notification.

For information on configuring IPN, see [Setting Up Instant Payment Notification \(p. 47\)](#).

## Errors

Amazon FPS error results provide information about syntactical errors in your requests, as well as errors that occur during the execution of your request (for example, a search that returns no results). Errors are returned only in response to REST requests. For SOAP requests, an error results in a SOAP fault.

In the Amazon FPS API Reference, each action description contains the list of errors that can be returned. For a list of all errors, see [Error Codes \(p. 55\)](#).

## REST Errors

If the original request to Amazon FPS used REST, in the case of an error, Amazon FPS returns an XML error response similar to the following. Errors consist of two elements: *code* and *message*.

```
Response : <?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Errors>
    <Error>
      <Code>InvalidTokenId_Sender</Code>
      <Message>Sender token is not valid.</Message>
    </Error>
  </Errors>
  <RequestID>67679d8a-fd87-4e44-b063-32a69bfc3c8b</RequestID>
</Response>
Response Code: 400>
```

The error code is a unique string that identifies the error; the error message is a human-readable description of the error. These elements are nested within an `Error` element. If a request generates more than one error, only the first error is reported.

Response codes are more generic errors of which the error code is a subset. For more information, see [Response Codes \(p. 23\)](#).

## SOAP Fault

If the original request to Amazon FPS used SOAP, in the case of an error, Amazon FPS returns a SOAP fault similar to the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:aws="http://webservices.amazon.com/AWSFault/2005-15-09">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>aws:Client.IncompatibleTokens</faultcode>
      <faultstring>The transaction could not be completed because the tokens
      have incompatible payment instructions:
```

```

Assertion Failed for Recipient</faultstring>
  <detail>
    <aws:RequestId xmlns:aws="http://webservices.amazon.com/
AWSFault/2005-15-09">
      ad56d51c-b1df-4b15-95ca-9f71c2c65eea
    </aws:RequestId>
  </detail>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

The elements `faultcode` and `faultstring` are the SOAP equivalents of the REST elements `Code` and `Message`, respectively.

## Response Codes

Amazon FPS returns response codes in three categories so that you can easily determine how best to handle a problem:

- **2XX**—Errors caused by mistakes in the request. For example, your request might be missing a required parameter. The error message in the response gives a clear indication of what is wrong.
- **4XX**—Errors that are transient  
These errors do not indicate a problem with Amazon FPS. So, upon receiving this error, resubmit the request.
- **5XX**—Errors that are non-transient  
These errors reflect problems with the underlying Amazon FPS web service. You will have to wait until the web service is functioning before resubmitting the request.

## CE and SE Status Codes

Amazon FPS returns a status code for each of the Co-Branded service requests you make. You can receive success and failure status codes for your requests. The status codes for each of the Co-Branded service APIs are listed in the respective topics in this guide. If you receive a caller exception (CE) or system error (SE) status code, you must handle them as described here.

### CE (Caller Exception)

A caller exception (CE) error code indicates that your Co-Branded service code has an error. We assume that you will encounter any caller exceptions when you test your Co-Branded service integration (before you go live). Therefore, when a caller exception occurs, Amazon FPS immediately displays on the screen an error message describing the problem, along with a **Continue** button. If you click **Continue**, the CBUI returns you (as the test buyer) to your web site (the return URL) and passes the caller exception error in the URI. You must fix the code that manages the requests to avoid receiving the error again.

Error Message	Description
CE - Caller Input Exception: The following input(s) are not well formed: [comma-separated list of input parameters]	The request parameters in the error message are not specified as mentioned in the request parameter description of the pipeline.
CE - Caller Input Exception: The following input(s) are either invalid or absent: [comma-separated list of input parameters]	The input parameters in the error messages are either incorrect or have not been specified in the request.

Error Message	Description
CE - Caller Input Exception: The following input(s) are not valid for this pipeline: [comma-separated list of input parameters]	The input parameters mentioned in the error messages should not be included for this pipeline. Please view the list of correct input parameters from the respective topic.

## SE (System Error)

A system error (SE) indicates that your Co-Branded service request has temporarily failed in Amazon FPS. You can retry the request again.

# Business Considerations

## Topics

- [Amazon Payments and Your Web Site \(p. 24\)](#)
- [Supported Payment Instruments and Currencies \(p. 24\)](#)
- [Amazon Payments Account \(p. 24\)](#)
- [Account Management \(p. 25\)](#)
- [Amazon Recipient Fees \(p. 25\)](#)
- [Fraud \(p. 25\)](#)
- [Disputes \(p. 25\)](#)

Running a business is more than just creating a web site. Creating a business involves creating policies and interacting with buyers. The business policies you make help determine the functionality you implement on your web site. This section discusses such business considerations.

## Amazon Payments and Your Web Site

You can add an Amazon Payments icon to your web site to let your buyers know you accept Amazon Payments. For more information, go to the [Marketing Toolkit](#).

## Supported Payment Instruments and Currencies

Amazon FPS supports the following payment instruments:

- Amazon Payments account balance (ABT)
- Bank account debits (ACH)
- Credit cards (Visa, MasterCard, American Express, Discover, Diners Club, and JCB)

Amazon FPS allows all Amazon.com customers (U.S. and international) to use major credit cards to make payments on Amazon Payments web sites. However, only US-based customers can use Amazon Payments account and bank account transfers. All transactions are conducted in U.S. dollars.

## Amazon Payments Account

If buyers already have an Amazon.com account, an Amazon Payments account is automatically created, and is activated when they make their first payment on any web site that accepts Amazon Payments.

If a buyer doesn't have an Amazon.com account, it's easy to create one: he or she only needs to supply an e-mail address and a password.

Buyers can also hold a monetary balance in their Amazon Payments accounts and use this money as a payment method just like a credit card or bank account. Buyers can manage their Amazon Payments accounts through the Amazon Payments web site.

## Account Management

Buyers, merchants, and developers can track transactions at <http://payments.amazon.com>. If you prefer to programmatically track transactions, you can use the Amazon FPS Account Management Quick Start implementation to get account information, for example, for a specified period. See the *Amazon FPS Account Management Quick Start Developer Guide*.



### Note

Buyers cannot see their account activity using their customer account on [www.amazon.com](http://www.amazon.com).

## Amazon Recipient Fees

Amazon Payments charges different fees for each of the different payment methods: credit cards, bank account debits, and Amazon Payments balance transfers. Amazon's cost to process a payment through a bank account debit is less than the cost via credit card. Amazon's cost to process an Amazon Payments balance transfer is less still. By exposing different fees for each of these three methods, Amazon Payments can pass on savings from bank account debits and balance transfers, allowing you to save money. In each case, Amazon Payments takes on the complexity of managing security and fraud protection. Fees are assessed on a per-transaction basis and vary depending on the payment method used and the transaction. For more information, go to the FAQ on the [Amazon FPS home page](#).

## Fraud

You can feel safe and secure while your customers shop on your web site. Amazon Payments is built upon Amazon's leading fraud protection technology. Under our Payment Protection Policy, we do not hold you liable for fraud-related chargebacks if you and the transactions meet all the requirements of the policy. You could still be held liable for service chargebacks. For details, go to our [User Agreement](#).

## Disputes

We want buyers to purchase with confidence when using Amazon Payments. However, disputes between buyers and merchants do occasionally occur. When this happens, buyers should first contact the merchant directly to try to find a solution. If the parties cannot resolve their dispute, the Amazon Payments Buyer Dispute Program provides a mechanism to address the buyer's complaint using the Amazon A-to-Z Guarantee.

When a buyer files a dispute, Amazon will notify the seller by e-mail. Based on the notification, the seller can choose to refund the transaction amount to the buyer or the seller can contest the dispute by providing details that prove of delivery of service or goods within 5 business days. Amazon FPS will resolve the dispute based on the information the buyer and the seller provide.

The seller should use the following tips to avoid disputes:

- Answer all buyer contacts (e.g., e-mails) promptly
- Be sure to deliver within the shipping estimate you provide
- Describe products accurately and provide clear images
- Keep buyers informed
- Work with buyers to resolve their negative order experiences
- Pick, pack, and ship securely. Don't skimp on packing
- Post a clear returns policy. Respond to return requests promptly with detailed instructions

- Promptly cancel any out of stock orders
- Refund as soon as possible when product defects or recalls become apparent

Amazon FPS does not provide actions to handle disputes. This section, however, addresses how to handle them.

## Amazon A-z Guarantee

The Amazon A-z Guarantee applies to qualified purchases of physical goods. Therefore, the following items are not covered by the Amazon A-z Guarantee: payments for services, digital merchandise, and cash equivalent instruments (including retail gift cards). The condition of the item purchased and its timely delivery are guaranteed under the Amazon A-z Guarantee. For transactions that are not covered by Amazon A-z Guarantee, the Amazon Payments Buyer Dispute Program still allows buyers to obtain assistance in seeking the merchant's further consideration of their complaint. Amazon Payments will attempt to resolve disputes by fostering good faith communication between buyers and merchants.

The item must be purchased from a merchant using Amazon Payments. The buyer must wait 15 days from the order date to submit a claim. From that point, the buyer has 90 days to submit a claim.

The Amazon A-z guarantee applies under the following conditions:

- If the item becomes defective more than 30 days past the shipment date and it is under warranty, the buyer must contact the manufacturer for repair or replacement. The buyer must provide all information required when submitting the claim.
- If the buyer paid by credit card, and the issuing bank has initiated a chargeback, the buyer is not eligible for coverage under the Amazon A-z Guarantee.

Buyers who pay for qualified physical goods using Amazon Payments are eligible to receive up to \$2,500 of the purchase price, including shipping charges.

Amazon has built up a base of millions of satisfied customers over the years through an intense focus on being responsive to their concerns and acting quickly to resolve any outstanding problems. The vast majority of customers never need to use the Amazon A-z Guarantee reimbursement program, but for those who do, the guarantee claim gives customers a greater sense of trust and confidence in shopping from the broad range of merchants.

## Amazon Buyer Dispute Program

The Amazon Buyer Dispute Program applies when the buyer has used Amazon Payments to purchase a non-physical item or service from a merchant; and either the buyer paid the merchant for the item or service but it did not arrive; or the buyer received the item, but the item is materially different than the way the merchant described it. For more information, go to [Buyer Dispute Program](#).

The A-z Guarantee only applies to the purchase of physical goods and does not apply to unlawful or prohibited items (including items violating the Amazon Payments Acceptable Use Policy or our User Agreement). For more information, go to the [Acceptable Use Policies](#) and [Amazon Payments User Agreement](#).

Buyers can submit a complaint by logging into their Amazon Payments account. For disputes involving physical goods that are covered under the Amazon A-z Guarantee, we will process a submission as an A-z Guarantee claim. Buyers also can submit an A-z Guarantee claim by viewing the specific transaction details via Your Account on the Amazon Payments web site. From the transaction or order details page, they can also click "**Problem with this transaction?**" or "**Problem with this order**" to file a claim.

Buyers can contact Amazon when the transaction has been resolved, but merchants are not able to withdraw claims filed by a buyer. Instead, if merchants believe that a pending claim should be revoked or canceled, they must contact buyers and encourage them to write to us. If the buyer and the seller

reach a resolution after a claim check was sent, we asks buyers to contact us to make arrangements for repayment.

## Chargebacks

A chargeback is a reversal of payment issued by the bank when a buyer disputes a charge. A chargeback can occur when a buyer has not received the items, has been charged multiple times for a single purchase, or is dissatisfied with the purchase and has not been able to resolve the matter with you. Chargebacks can happen only with credit card transactions.

Typically, a buyer contacts his or her bank to request a chargeback. The bank notifies the credit card association, which in turns notifies us. We work with the credit card company to resolve the chargeback. We may request information from you to dispute the chargeback with the credit card association.

Amazon FPS works with you and the buyer to resolve the chargeback. You have 5 business days to respond to the chargeback notification Amazon FPS sends you and to supply any requested information. If you do not respond within this time period, the dispute is automatically granted to the buyer.

Use the following tips to avoid chargebacks:

- Charge buyers once for a single order to avoid duplicate billing  
If you receive two or more identical orders, verify the information with the buyer
- Avoid dissatisfaction with item quality by providing a detailed description of items on your web site, including specifications, measurements, and capabilities  
Other aids such as audio, video, photographs, or drawings are also helpful
- Make the shopping experience positive for your buyers:
  - Provide help when your buyers have questions or need assistance
  - Clearly explain to your buyers when their order will ship and keep them informed about the progress of their orders
  - Make sure that items are delivered promptly without damage
  - Ship items with carriers who provide online item tracking and require signatures on delivery
  - Respond promptly to e-mail from your buyers
  - Publish your policies for cancellations and returns to avoid chargebacks
  - Refund an order when it is necessary to do so

## WSDLs and Schemas

Web services involve the exchange of requests and responses between computers communicating over the Internet. So that computers running different operating systems can communicate, the vocabulary for the communication must be established. A WSDL is a dictionary of terms that two computers can use to structure requests and responses. Schemas typically contain type definitions of the terms in the WSDL.

This section provides a brief introduction to WSDLs and schemas and also provides the location for the Amazon FPS WSDL and schema.

## WSDL

A WSDL (Web Service Description Language) is an XML document that defines the operations, parameters, requests, and responses used in web service interactions. You can think of a WSDL as the contract that defines the language and grammar used by web service clients and servers. When you look at the Amazon FPS WSDL, for example, you find in it all of the Amazon FPS operation names, parameters, request and response structures.

There is not a single WSDL. Amazon FPS, for example, has many different versions of its WSDL—the latest one and all of its previous versions. Not only can one company use different versions of a WSDL, every company can use its own WSDL based on its own APIs or business metrics. For that reason, web service requests must identify the WSDL they use so the web servers know how to interpret the requests.

The latest Amazon FPS WSDL is at: <https://fps.amazonaws.com/doc/2008-09-17/AmazonFPS.wsdl>.

## Schema

A schema is similar to a WSDL in that both are XML documents. Whereas the WSDL defines the web service language used by computers to converse, the schema defines the data types used in the WSDL.

You do not have to create schemas to use Amazon FPS. Those have already been created. It is helpful, however, to understand schemas so that you can determine the data types returned in responses.

The W3C defines the base data types, which include, for example, int, string, and float. While these data types are useful, they are not very descriptive. For example, defining every occurrence of text in an XML document as being of type string hides the differences between text that might be, for example, a paragraph versus a note. In such an application where paragraphs and notes are used, a schema would contain an extension of the string base class so that paragraph (<para>) and note (<note>) could be used as tags in XML documents.

The latest Amazon FPS schema is at: <https://fps.amazonaws.com/doc/2008-09-17/AmazonFPS.xsd>. A graphic representation is available at <http://docs.amazonwebservices.com/AmazonFPS/latest/SchemaDocs/fps-schema.html>.



# Programming Guide

## Topics

- [Important Values to Store in Your Database \(p. 30\)](#)
- [Getting Authorization \(p. 30\)](#)
- [Recipient Registration \(p. 31\)](#)
- [Making Payments \(p. 33\)](#)
- [Handling Transactions that Don't Return \(p. 36\)](#)
- [Order Cancellations \(p. 37\)](#)
- [Testing Your Applications for Free \(p. 38\)](#)
- [Working with Signatures \(p. 41\)](#)
- [Soft Descriptor Customization \(p. 45\)](#)
- [Setting Up Instant Payment Notification \(p. 47\)](#)

The Programming Guide provides task-oriented descriptions of how to use and implement Amazon Flexible Payments Service (FPS) actions. For a complete description of Amazon FPS actions, see the [Amazon FPS API Reference \(p. 53\)](#).

The following table describes the topics discussed in the programming guide.



### Note

To perform these tasks, you must have an Amazon FPS developer account. For information about getting the account, go to [Amazon Flexible Payments Service Getting Started Guide](#).

If you want to...	Read this section...
Enable buyers to authorize payments	<a href="#">Getting Authorization (p. 30)</a>
Enable merchants to register on your web site so they can receive payment for their items sold through your web site	<a href="#">Recipient Registration (p. 31)</a>
Process the buyers' payments	<a href="#">Making Payments (p. 33)</a>
Cancel an order or refund a payment	<a href="#">Order Cancellations (p. 37)</a>

If you want to...	Read this section...
Test your application using the Amazon FPS sandbox	<a href="#">Testing Your Applications for Free (p. 38)</a>
Use the Amazon FPS Sample Code	<a href="#">Code Samples (p. 125)</a>
Create request signatures	<a href="#">Working with Signatures (p. 41)</a>
Get notifications about transactions	<a href="#">Setting Up Instant Payment Notification (p. 47)</a>

## Important Values to Store in Your Database

When you use Amazon FPS, there are times when you should store important information in your database. The following sections describe some important values you should store.

### Caller Reference

The *CallerReference* is a string you provide that uniquely identifies a request. An appropriate value to use is the order ID. You can also use the value to retrieve information about a transaction or to retrieve the related token (for more information, see [Co-Branded Service Requests that Don't Return \(p. 36\)](#)). Amazon FPS uses the caller reference value to provide request idempotency for a seven-day period (for more information, see [Resending Requests \(p. 36\)](#)).



#### Note

If you perform multiple partial refunds for a particular payment, you must provide a different caller reference value for each partial refund request.

### Transaction ID

The transaction ID is a string Amazon FPS creates to uniquely identify each transaction in the FPS system. The Co-Branded service doesn't return a transaction ID; only Amazon FPS does (e.g., in a `Pay` response). You should maintain the transaction ID in your database and associate it with your caller reference value for the order. Because of network issues, it's possible that the response to your `Pay` call might not reach you, so you won't have a transaction ID to store in your database. In that case you can resend the original request (within 7 days) and receive the response again (for more information, see [Resending Requests \(p. 36\)](#)).

### Request ID

Amazon FPS returns a request ID for each Amazon FPS API call accepted for processing. If you have a problem with a request, AWS asks for the request ID to troubleshoot the issue.

## Getting Authorization

Before you can issue an Amazon FPS request that charges a buyer for an item, you must get the buyer's authorization. The authorization process uses the Amazon Co-Branded service, which has a different API from the Amazon FPS web service. Some of the values returned by the Co-Branded web service, however, are required in Amazon FPS requests. When the buyer authorizes a purchase, the Amazon Co-Branded service creates a payment token, which enables the exchange of money from buyer to seller.

This section describes how to use the Co-Branded API and payment tokens. The remainder of this guide describes how to use the Amazon FPS API.

## Sending a Co-Branded Service Request

This section shows how to send a request that redirects the buyer to the CBUI. You must send a Co-Branded service request before you can use an Amazon FPS `Pay` or `Reserve` request.

These requests are typically implemented as an HTML form on your web site. Your site dynamically updates the values of the Co-Branded service request parameters according to the items purchased.

### To send a Co-Branded service request

1. Add up all the charges for all of the items the buyer wants to purchase, together with all taxes, shipping fees, and any additional fees (such as gift wrapping fees).
2. Use the Recipient Token API, the Recurring-Use Token API, or the Multi-Use Token API, depending on the kind of payment token you want to create (for more information about the APIs, see [Co-Branded Service API Reference](#) (p. 109)). For a list of the parameters common to all Co-Branded service requests, see [Common Parameters](#) (p. 109).

This example request is for a recurring payment for the download of a number of songs per month.

```
https://authorize.payments.amazon.com/cobranded-ui/actions/start?
  callerKey=[The caller's AWS Access Key ID]
  &callerReference=DigitalDownload1183401134541
  &paymentReason=Monthly+download+subscription
  &pipelineName=Recurring
  &recurringPeriod=1+Month
  &returnURL=http%3A%2F%2Fwww.digitaldownload.com%2FpaymentDetails.jsp
%3FPaymentAmount%3
D05.00%26Download%3DMonthlySubscription%26uniqueId%3D1183401134535
&signatureVersion=2
&signatureMethod=HmacSHA256
&signature=[URL-encoded value you generate]
&transactionAmount=5.00
```

The optional parameters vary in your request according to what the buyer purchases.

For information about getting your AWS Access Key ID value, go to the [Amazon Flexible Payments Service Getting Started Guide](#).

3. Programmatically populate this request with the parameter values based on the items the buyer is purchasing.
4. Calculate the signature and include it in the request.  
For more information about creating the correct value for *signature*, see [Working with Signatures](#) (p. 41).
5. Implement the `Pay Now` button on your web site to send this request.



### Note

We recommend that you first try your Co-Branded service request in the Amazon FPS Sandbox. For more information, see [Testing Your Applications for Free](#) (p. 38).

## Recipient Registration

As a caller offering marketplace services, you must provide a way to register recipients so that they can use your services, which include uploading product information for you to display on your web

site and facilitating the sale of this merchandise. You must collect enough information to identify each recipient so they can get paid for products purchased by buyers. You must handle the upload of product information to your web site. Amazon FPS Advanced Quick Start can handle the registration of your recipients and the exchange of money from sender to recipient.

The actions `Pay` and `Reserve`, which initiate payment transactions, require parameter values that identify the sender and the recipient. In the API, these identifiers are called tokenIDs. There is a `SenderTokenId`, which identifies the sender (who sends the money), and a `RecipientTokenId`, which identifies the recipient (who receives the money). Because you, the caller, send `Pay` requests on behalf of the others, you must obtain those identifiers.

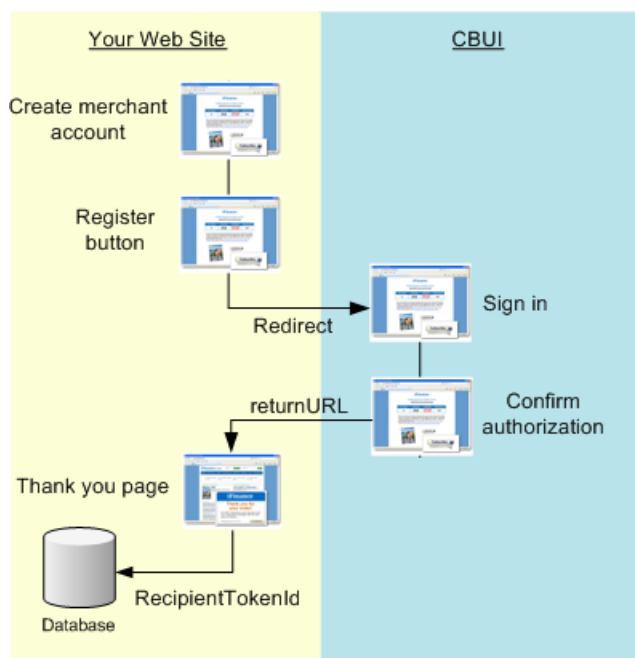
These identifiers are generated by the Amazon Co-Branded service at the request of the Co-Branded User Interface code (CBUI), which is a series of web pages that, among other things, identifies the recipient or sender. This means that your web site must have an interface that sends a Co-Branded service request for the sender and a different one for the recipient. The procedures for sending both requests are the same; the only difference is that they use different Co-Branded service APIs. The API used for recipient requests returns a `RecipientTokenId`, and the API used for the sender requests returns a `SenderTokenId`.

This section describes how to use the Co-Branded service to register the recipient.

## Recipient Registration on Your Web Site

The following figure represents the first step in the workflow: recipient registration on your web site. Merchants must register with you for the following reasons:

- The recipient must accept your business terms, in particular, the marketplace fee you will charge them.
- The recipient must be able to upload and otherwise manage the item information to your web site.
- You must have a `RecipientTokenId` for each recipient so that you can pay them using that parameter in a `Pay` or `Reserve` request.



## Recipient Registration

1	Implement a recipient account system on your web site. For example, enable each recipient to sign in to their account with a sign-in name and password.
2	As the final stage of your registration process, implement a <i>Register</i> button to send a recipient Co-Branded service request. For more information, see <a href="#">Recipient Token API (p. 112)</a> .
3	The recipient enters the required information on the Amazon CBUI pages.
4	After confirming his or her choices, the recipient is redirected back to your web site with information, such as the <i>RecipientTokenId</i> that you should store in your database.

## Implementing the Co-Branded API to Register a Recipient

The Co-Branded API you use for registering a recipient is the Recipient Token API. For more information, see [Recipient Token API \(p. 112\)](#).

### To register a recipient

1. On your web site, enable a recipient to register with you.

In this step you collect information about the person or company using your services.

2. Direct each recipient to <https://payments.amazon.com/sdui/sdui/premiumaccount> to create a business payment account so they can get paid.
3. On your web site, display your business policies, including your marketplace fee structure, and obtain the recipient's acknowledgment.

Your marketplace fee might include a flat fee, a percentage of the purchase price, or both. You implement your fee structure using the parameters in the Co-Branded service request.

4. Implement a button that issues a Co-Branded service request that registers the recipient.

For more information, see [Sending a Co-Branded Service Request \(p. 31\)](#).

5. Parse the response.

In particular, store the *tokenID*, which is the recipient's. You use this value in *Pay* and *Reserve* requests. You also need to store the *RefundTokenID* to use in case you need to refund a future transaction.

## Making Payments

### Topics

- [Transacting the Payment \(p. 34\)](#)
- [Failed Payment Transactions \(p. 35\)](#)
- [Changing the Payment Instrument \(p. 35\)](#)
- [Notifications \(p. 36\)](#)

After the sender authorizes the purchase on the CBUI web pages, the URI returned contains a successful *status* value. Upon receiving this response, you must send a *Pay* request to


actually initiate the transfer of money from the sender to the recipient. In the Advanced Quick Start implementation, the recipient could be you or a merchant whose e-commerce site you host.

This section describes how to make those payments.

## Transacting the Payment

The sender uses the [CBUI \(p. 15\)](#) to authorize the payment. Upon the successful authorization, the CBUI redirects the sender to the URL specified by the `returnURL` parameter in the CBUI request. When you parse this returned URI, you check the `status` parameter and `returnURL`, among other values. If the status is one of the success values, you need to send a `Pay` request to start the purchase transaction. A successful `Pay` request immediately charges the sender's payment instrument, such as a credit card. `Pay` can accept all payment instrument types, including credit card, bank account debit, and Amazon Payments withdrawal.

### To transact a payment using Pay

1	<p>Obtain the list of items being purchased by the sender from your web site and derive the values required for the parameters in the <a href="#">CBUI (p. 15)</a> request.</p> <p>One of the parameters is <code>TransactionAmount</code>. The value for this is the total charge to the sender, including tax, shipping and any other fees. Another parameter is <code>CallerReference</code>. You generate this identifier and associate it with the payment instrument created. You can index the transactions on your database based on its value. This way, you can match the identifier on your database, <code>CallerReference</code>, to the identifier on the Amazon FPS database, <code>TransactionId</code>. For more information about the parameters, see <a href="#">CBUI (p. 15)</a>.</p>
2	<p>Implement on your web site the equivalent of a <i>Pay Now</i> button so that it sends the CBUI request.</p> <p>For more information about constructing the CBUI request, see <a href="#">CBUI (p. 15)</a>. When the sender authorizes the payment in the CBUI, the sender is redirected to the URL specified by the <code>returnURL</code> parameter in the CBUI request. This URL is typically a <i>Thank you</i> page and one that invites the sender to keep shopping.</p>
3	<p>Parse the returned URI.</p> <p>In addition to the URL specified by <code>returnURL</code>, the URI contains parameters added by the CBUI. Those parameters include all of the parameters in the CBUI request, which are helpful for debugging purposes, a <code>TokenId</code>, which you must subsequently submit with the <code>Pay</code> request, and a <code>status</code>, which tells you whether or not the sender successfully authorized the payment.</p>
4	<p>If the <code>status</code> value is a success value, programmatically submit a <code>Pay</code> request. The required parameters include <code>SenderTokenId</code>, which maps to the <code>TokenId</code> you received in the CBUI response, <code>TransactionAmount</code>, which you entered in the CBUI request, and <code>CallerReference</code>, which is a value you generate that uniquely identifies the <code>Pay</code> transaction.</p> <p> <b>Note</b></p> <p>The CBUI API uses the parameter, <code>CallerReference</code>, and the Amazon FPS API uses <code>CallerReference</code>. It is easier if you make both parameters equal the same value. We recommend that you make the <code>CallerReference</code> the same as the order ID on your website.</p>

5	Parse the response. Two important values the <code>Pay</code> request returns are <code>TransactionStatus</code> , which tells whether the charge was successful against the sender's payment instrument, and <code>TransactionId</code> , which is the identifier Amazon Payments uses to identify this transaction. You should maintain the <code>TransactionId</code> in your database and associate with your <code>CallerReference</code> value for the transaction.
6	Upon a successful transaction, add a task to your workflow to fulfill the order.

## Failed Payment Transactions

There can be times when Amazon Payments charges a sender's payment instrument and that transaction fails. There are two kinds of failure:

- **Failure**—A transaction is canceled for non-payment reasons.  
For example, a transaction might be considered fraudulent and is therefore refused.
- **Hard decline**—A financial institution refuses the transaction.  
This could happen when a credit card has exceeded its maximum limit or when a credit card has expired. There is no retry after a hard decline.

You do not need to take any action in either of these cases. In the case of a failed transaction, Amazon Payments e-mails the sender and you about the transaction decline.

## Repeated Pay Requests

Due to network problems, some `Pay` requests might not complete successfully. If this happens, it might be possible to recapture the information and resend the request. For more information, see [Resending Requests](#) (p. 36).

## Changing the Payment Instrument

It's possible that before a recurring or multi-use payment token expires, the sender must change the payment instrument associated with the token. This might happen, for example, when a credit card expires or is otherwise terminated. To handle this issue, you must enable the sender to change the payment instrument for an ongoing payment token. You do that by sending a Co-Branded service request for the Edit Token API, as described in the following process.

### Process for Changing a Payment Token's Payment Instrument

1	Enable the sender to discover on your web site the ongoing payment token that needs modification.
2	Enable the sender to enter the new payment instrument data, such as a credit card number.
3	Use the input to construct a Co-Branded service request for the Edit Token API. For more information, see <a href="#">Sending a Co-Branded Service Request</a> (p. 31) and <a href="#">Edit Token API</a> (p. 122).
4	Implement the equivalent of a <i>Change Payment Method</i> button to issue the Co-Branded service request.

5	Parse the returned URI. In addition to the URL specified by <i>returnURL</i> , the URI contains parameters added by the Co-Branded service. Those parameters include all of the service response parameters, including <i>status</i> , which tells you whether or not the sender successfully changed the payment instrument. Part of the job of parsing the URI is evaluating the <i>signature</i> parameter to make sure the response has not been altered. For more information, see <a href="#">Verifying the ReturnURL and IPN Notifications</a> (p. 43).
6	Display an appropriate web page to the sender based on the value of the <i>status</i> parameter.

## Notifications

If the sender uses an Amazon Payments account balance to make the purchase, the success or failure of the transactions occurs quickly. If, however, the sender uses a bank account withdrawal or a credit card to make the purchase, the results often take a while to complete. To be notified of the status of the transaction, you must create a web service that receives Instant Payment Notification (IPN) updates, and enable IPN updates. For more information, see [Setting Up Instant Payment Notification](#) (p. 47).

## Handling Transactions that Don't Return

When a customer buys a product on your web site, their expectation is that the product will be paid for and delivered, but sometimes problems cause service requests to become lost. In those cases, it is sometimes possible to find the lost transaction.

### Co-Branded Service Requests that Don't Return

After the sender finishes the CBUJ pages and authorizes the payment, the service should redirect the sender to the URL you specified in the *returnURL* parameter in the CBUJ request. There might be times, however, when the sender authorizes the payment and the redirect fails. Because you don't have the *SenderTokenId*, you can't charge the sender for the authorized purchase. Use the following procedure when the redirect from service fails.

#### To recover the SenderTokenId

1. Query your database to get the *CallerReference* you supplied in the Co-Branded service request for the transaction in question.
2. Send a *GetTokenByCaller* request with the *CallerReference*.
3. If there isn't an *Errors* element in the response, the sender authorized the payment but there was some problem with the redirect to *returnURL*. The response includes *SenderTokenId*.  
If there is an *Errors* element in the response, the authorization did not succeed.

### Resending Requests

There are times when network problems prevent the completion of requests. This could happen both for Co-Branded service requests and for FPS API requests. Your application should periodically check for this condition, and if it occurs, retry the request.



## Co-Branded Service Requests

To check for Co-Branded service requests that didn't complete, whenever you send a request, put the caller reference value from the request in your database. When you get the response, store the token ID that you receive against the caller reference in your database.

About once each hour, resend any requests that don't have a sender token ID stored against the caller reference. The timing is important, because for most tokens, Amazon FPS maintains the token IDs for only three hours.

## Amazon FPS API Requests

Each time you send a request to the Amazon FPS API, FPS maintains the caller reference from the request for 7 days and uses it to check for duplicate requests. If you don't receive a response to an Amazon FPS API request, you can resend the exact same request within that seven-day period, and Amazon FPS will return the original response and not create a new transaction. If the first request succeeded, the second request does not charge the sender's payment instrument a second time.

In that seven-day period, if you send a request with that same caller reference value but other parameter values, Amazon FPS returns a `DuplicateRequest` error.

After the seven-day period, if you send the original request again (with the same caller reference and parameter values), Amazon FPS creates a new transaction.

# Order Cancellations

### Topics

- [Canceling a Recurring Transaction \(p. 37\)](#)
- [Refunding a Recurring Transaction \(p. 38\)](#)
- [Other Reversals and Issues \(p. 38\)](#)

There are times when the sender decides to cancel a recurring payment. For example, the sender might decide in the middle of a subscription to stop receiving a newspaper or magazine. This chapter describes how to cancel recurring transactions.

## Canceling a Recurring Transaction

The Amazon Payments web site, [payments.amazon.com](https://payments.amazon.com), enables senders to cancel recurring transactions from their user account. To provide transaction cancellation functionality on your web site, use the following procedure.

### To cancel a recurring transaction

1. Enable the sender to find the recurring transaction to cancel.
2. Obtain from that transaction the `TokenId`, which was returned in the Co-Branded service response.
3. Submit a `CancelToken` request with the token ID.

For more information about this action, see [CancelToken \(p. 65\)](#).

The following `CancelToken` request cancels the recurring transaction specified by the `TokenId`.

```
https://fps.sandbox.amazonaws.com?  
Action=CancelToken
```

```
&AWSAccessKeyId=AKIAIUQNNI2DNQHBO7RA
&ReasonText=MyWish
&Signature=IZD9O%2FWGqhkzO%2FdLTQ7Tn8KUAmtZXqIEg6gypwkGeWQ%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-07T08%3A46%3A37.156Z
&TokenId=D739IT9TMC4FK9KB56PDKJWAQGXDX3B8X3SJNGVH3UEF5GQ7XAQZMEIL4OGEZKGX
&Version=2008-09-17
```

## Refunding a Recurring Transaction

Amazon FPS gives you the option of refunding an unused portion of a recurring payment that the sender paid for but will not use. This might happen if, for example, the sender pays membership fees twice annually but must cancel only three months into the membership. The `Refund` action has an optional parameter, `RefundAmount`, which enables you to refund all or only a portion of the transaction amount a sender paid.

The Amazon Payments web site, [payments.amazon.com](http://payments.amazon.com), enables senders to refund transactions from their user account. To provide transaction refunding functionality on your web site, use the following procedure.

### To refund a transaction

1. Enable the sender to find the transaction to refund.
2. Obtain from that transaction either the `TransactionId` or the `CallerReference`.
3. Submit a `Refund` request using one of those values.

For more information about this action, see [Refund \(p. 81\)](#).

The following `Refund` request refunds the transaction specified by the `CallerReference` and `TransactionId`.

```
https://fps.sandbox.amazonaws.com?
Action=Refund
&AWSAccessKeyId=AKIAIIFXJCFIHITREP4Q
&CallerDescription=MyWish
&CallerReference=CallerReference03
&RefundAmount.CurrencyCode=USD
&RefundAmount.Value=1
&Signature=V6pU3PvDPkPhR9Eu7yZXnFZHueFafLE5sBPgqqCELEU%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T05%3A51%3A49.578Z
&TransactionId=14GK4TNCAQ84NK9VITEHKAS94RAD9ZE2AQD
&Version=2008-09-17
```

## Other Reversals and Issues

In the e-commerce world, there are other types of transaction reversals, including chargebacks and claims. Amazon Payments does not provide actions to handle those specific activities. So, those implementations are left to you on your web site. For more information, see [Business Considerations \(p. 24\)](#).

# Testing Your Applications for Free

### Topics

- [Sandbox Endpoints](#) (p. 39)
- [Sandbox Use](#) (p. 39)
- [Error Simulation](#) (p. 39)
- [Testing Signatures](#) (p. 40)

Amazon FPS provides a sandbox environment that you use to test your applications. In the sandbox you can try out your applications without incurring charges or making purchases. We recommend that you test all of your requests in the sandbox before exposing them on your web site.

The Amazon FPS Sandbox enables you to:

- Make Amazon FPS web service and Co-Branded service requests
- Make `Pay` requests to transfer money
- Use credit cards and bank accounts in your test transactions without any prior verification and without incurring charges
- Simulate errors

You can simulate certain errors that could appear in a real transaction. This simulation can help you test the error handling capabilities in your application.

For information about signing up for an Amazon FPS Sandbox account, go to the [Amazon Flexible Payments Service Getting Started Guide](#). For more information about the Amazon FPS Sandbox, go to <https://payments-sandbox.amazon.com>.

## Sandbox Endpoints

Sandbox endpoints are different from Amazon FPS production endpoints. The Amazon FPS Sandbox endpoints are as follows:

- **Amazon FPS API**—<https://fps.sandbox.amazonaws.com>
- **Amazon Co-Branded service**—<https://authorize.payments-sandbox.amazon.com/cobranded-ui/actions/start>
- **Manage Sandbox Account**—<https://authorize.payments-sandbox.amazon.com>
- **Central FPS Sandbox Resource page**—<https://developer.payments-sandbox.amazon.com/landingpage>

## Sandbox Use

You can test the following user experiences in the sandbox:

- Registering for a business or personal account via a Co-Branded service request
- Depositing funds into a test account's Amazon Payments account using a `Pay` request
- Checking the account balance for a test account
- Checking the activity for a test account

## Error Simulation

The sandbox accepts any random number as a credit card and token ID in `Pay` and `Reserve` requests. However, you can simulate a variety of declines that occur by using specific token IDs and amounts in the Amazon FPS Sandbox, as shown in the following tables.

The following table shows the errors you can simulate by entering specific *SenderTokenId* values.

Error	SenderTokenId Value
Closed account	Z1LGRXR4HMDZBSFKXELA32KZASGWD8 IHMHZ CK4DETR784LDLD1GMFW4P3WT8VTGX
Email address not verified	E3FR7BARJV3PB631PMKV74PGKCJLBHI1Q1K MQN7BJ2JJICPDKN3N1CJIKFZ8D7NN
Suspended account	H216UECZ8ZM1G8G4QA3V7RKF8JDFZ9SI3SJA FSGUKBBNDHX1NVM8GUQRZNRNAHER

The following table shows the errors you can simulate by entering specific *RecipientTokenId* values. These token IDs are relevant only in marketplace environments.

Error	RecipientTokenId Value
Closed account	P1LL7A1LHK935DBGI5NAYCXOCLVEBHBNIU 7PBXBAMRKKNLDEPI8M3MUSLZT2VANZ
Email address not verified	C4LGSEMKN11FTUXZ2X2C7QVFHN5DVBGQJ NF17AIQXXXQSX4DRG4KJFCN2KRFFUZZI
Suspended account	R3VK49XVGCAZTJSXKN7ZSBHPMFGKM5VEEQTX GMVE8CFUZ2G5RLLMAB4J6TQRL6BU

With the Amazon Payments developer sandbox, you can force an error by placing certain decimal values in the amount. The following table details the values.

Force Condition	Error Forced	Simulation
The amount includes a decimal value between .60 and .69	Temporary Decline	Occurs when a downstream process is not available.
The amount includes a decimal value between .70 and .89.	Payment Error	Insufficient funds



**Note**

If you want your test transaction to be a success, avoid using amount values which contain decimal values between .60 and .89. For example, the following amounts all force errors: 0.61, 123.6522, 1.79. The following amounts do not force an error: 0.16, 123.56, 8.97.

## Testing Signatures

You can easily test your signature creation code using any of the examples in [Amazon FPS API Reference \(p. 53\)](#). Each example contains a signature calculated from the values in the rest of the example.

1. Copy any one of the sample query request examples from among the Actions in [Amazon FPS API Reference \(p. 53\)](#).
2. Remove the HTTP verb (GET or POST) and the URI from your copy. Also remove the explicit '\n' characters.

3. Remove the line with the *Signature* parameter from your copy.
4. Create a signature using the instructions in [Generating a Signature \(p. 42\)](#)
5. Compare the output from your signature creation code with the value you removed from the HTML example. They should be identical.

## Migrating your Application from the Sandbox to Production

When your application is running correctly in the sandbox, you need to do the following to switch it to the production environment:

### Launch Process

1	Change the Amazon FPS sandbox endpoint to the Amazon FPS live endpoints as listed in the following table:								
	<table border="1"><thead><tr><th>Application</th><th>Endpoint</th></tr></thead><tbody><tr><td>Co-Branded UI Pipeline</td><td><a href="https://payments.amazon.com/sdui/sdui/managecobranding">https://payments.amazon.com/sdui/sdui/managecobranding</a></td></tr><tr><td>Amazon FPS web service requests</td><td><a href="https://fps.amazonaws.com/">https://fps.amazonaws.com/</a></td></tr><tr><td>Amazon Payments Account Management UI</td><td><a href="https://payments.amazon.com/">https://payments.amazon.com/</a></td></tr></tbody></table>	Application	Endpoint	Co-Branded UI Pipeline	<a href="https://payments.amazon.com/sdui/sdui/managecobranding">https://payments.amazon.com/sdui/sdui/managecobranding</a>	Amazon FPS web service requests	<a href="https://fps.amazonaws.com/">https://fps.amazonaws.com/</a>	Amazon Payments Account Management UI	<a href="https://payments.amazon.com/">https://payments.amazon.com/</a>
Application	Endpoint								
Co-Branded UI Pipeline	<a href="https://payments.amazon.com/sdui/sdui/managecobranding">https://payments.amazon.com/sdui/sdui/managecobranding</a>								
Amazon FPS web service requests	<a href="https://fps.amazonaws.com/">https://fps.amazonaws.com/</a>								
Amazon Payments Account Management UI	<a href="https://payments.amazon.com/">https://payments.amazon.com/</a>								
2	If your application is set up to receive IPN notifications, set its IPN URL at <a href="https://payments.amazon.com/sdui/sdui/managecobranding">https://payments.amazon.com/sdui/sdui/managecobranding</a> .								
3	For a marketplace application, make sure you register for that option when you register the application in the production environment. If you did not select the option when you registered the application, you can file a contact-us request at <a href="https://payments.amazon.com/sdui/contactus">https://payments.amazon.com/sdui/contactus</a> .								
4	Please ensure you have specified your co-branding URL on production using the form at <a href="https://payments.amazon.com/sdui/sdui/managecobranding">https://payments.amazon.com/sdui/sdui/managecobranding</a> .								
5	Please ensure that the rest of your account settings are current at <a href="https://payments.amazon.com/sdui/sdui/accountsettings">https://payments.amazon.com/sdui/sdui/accountsettings</a> .								

You can use the same credentials to sign your requests as long as your Amazon Payments Developer account on both Sandbox and Production are linked to the same e-mail address and password.

## Working with Signatures

### Topics

- [Generating a Signature \(p. 42\)](#)
- [Verifying the ReturnURL and IPN Notifications \(p. 43\)](#)

- [Access Key Rotation \(p. 45\)](#)

This section provides detailed explanations for some of the tasks required to generate a signature. A signature is required for every request. For sample code for generating signatures, see [Code Samples \(p. 125\)](#).

## Generating a Signature

Web service requests are sent using SSL (HTTPS) across the Internet and are subject to tampering. Amazon FPS uses the signature to determine if any of the parameters or parameter values were changed in a web service request. Amazon FPS requires a signature to be part of every request.

### To create the signature

1. Create the canonicalized query string that you need later in this procedure:
  - a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when `Content-Type` is `application/x-www-form-urlencoded`).
  - b. URL encode the parameter name and values according to the following rules:
    - Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen ( - ), underscore ( \_ ), period ( . ), and tilde ( ~ ).
    - Percent encode all other characters with `%XY`, where X and Y are hex characters 0-9 and uppercase A-F.
    - Percent encode extended UTF-8 characters in the form `%XY%ZA...`
    - Percent encode the space character as `%20` (and not `+`, as common encoding schemes do).



#### Note

Currently all AWS service parameter names use unreserved characters, so you don't need to encode them. However, you might want to include code to handle parameter names that use reserved characters, for possible future use.

- c. Separate the encoded parameter names from their encoded values with the equals sign ( = ) (ASCII character 61), even if the parameter value is empty.
  - d. Separate the name-value pairs with an ampersand ( & ) (ASCII code 38).
2. Create the string to sign according to the following pseudo-grammar (the "`\n`" represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +  
ValueOfHostHeaderInLowercase + "\n" +  
HTTPRequestURI + "\n" +  
CanonicalizedQueryString <from the preceding step>
```

The `HTTPRequestURI` component is the HTTP absolute path component of the URI up to, but not including, the query string. If the `HTTPRequestURI` is empty, use a forward slash ( / ).

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key as the key, and SHA256 or SHA1 as the hash algorithm.  
For more information, go to <http://www.ietf.org/rfc/rfc2104.txt>.
4. Convert the resulting value to base64.
5. Use the resulting value as the value of the `Signature` request parameter.



### Important

The final signature you send in the request must be URL encoded as specified in RFC 3986 (for more information, go to <http://www.ietf.org/rfc/rfc3986.txt>). If your toolkit URL encodes your final request, then it handles the required URL encoding of the signature. If your toolkit doesn't URL encode the final request, then make sure to URL encode the signature before you include it in the request. Most importantly, make sure the signature is URL encoded *only once*. A common mistake is to URL encode it manually during signature formation, and then again when the toolkit URL encodes the entire request.

## About Signature Version 2

For *inbound requests*, signature version 2 signing uses the entire request uri as the basis for the signature, and encryption is based on the unique security credentials for your account.

For *outbound notifications*, signature version 2 provides two ways to verify that return URL responses and IPN notifications originate from Amazon Payments:

- A new Amazon FPS action, *VerifySignature*, enables you to securely check a response using a server-side call. This is the simplest and recommended way. For more information, see [VerifySignature](#).



### Note

You don't have to sign the *VerifySignature* request, nor do we require you to have a developer account to invoke *VerifySignature*.

- Both the return URL and IPN responses also include a *certificateUrl* parameter, which contains a URL to a signing certificate. If you prefer to validate the signature on the client side, you can use the certificate for validating the response. (The certificate is cached on your server and automatically updated when needed.) For more information, see [Client-side Signature Validation \(p. 44\)](#).

For *inbound requests*, signature version 2 supports AWS [access key rotation](#), further enhancing the security of your button content. For more information, see [Access Key Rotation \(p. 45\)](#).



### Important

The previous method for signing will expire on 01 November, 2010. At that time, any signing you do with your access keys must be done using the new method.

## Verifying the ReturnURL and IPN Notifications

Amazon Simple Pay sends you [outbound notifications](#) for both the ReturnURL and IPN notification. For the ReturnURL, it is in the form of GET data, and for IPN notification, it is POST data. When you handle these notifications, we recommend you validate the signature to ensure the notification originated from Amazon Payments. The signature version 2 security has two methods for you to verify the signature of the response:

- Server-side signature verification using the [VerifySignature \(p. 93\)](#) FPS Action. This is the method we recommend for Amazon FPS . To use it, modify your *returnUrl* and *ipnUrl* pages to parse the notification. From those components, you assemble the relevant parameters for *VerifySignature*. The result of the call from is either *Success*, meaning the response is valid, or *Failure*, indicating the response is suspect.

For more information on *VerifySignature*, see [VerifySignature \(p. 93\)](#). In addition, you can use the validation samples to assist creating your own validation pages. For more information, see [Return URL Validation Sample](#).

- Client-side signature verification using [PKI](#). This method is intended for applications which need to process a greater number of transactions. The notifications sent to both the `returnUrl` and `ipnUrl` endpoints contain the `certificateUrl` parameter. The `certificateUrl` value is a URL which specifies the location of the certificate used for signing the response. (For optimum performance, you can download the certificate and cache it locally. It only needs to be updated on its annual expiration date). You then use the certificate to validate the request using PKI. For more information, see [Client-side Signature Validation](#) (p. 44).

## Client-side Signature Validation




### Important

If you are using signature version 1 to validate the notifications from Amazon Payments, we strongly recommend you convert to signature version 2. For more information, see [Appendix: Moving your Application to Signature Version 2](#) (p. 139)

The following process describes how to verify the legitimacy of an *outbound notification* from Amazon Payments. You use the same process for Return URL and IPN posts..)

### Validating the signature in the Return URL and IPN notifications

1	Decode the signature in the notification.
2	Decode and read the <code>signatureVersion</code> and <code>signatureMethod</code> parameters from the notification. The value of <code>signatureVersion</code> value should be 2, and the value for <code>signatureMethod</code> value should be <i>RSA-SHA1</i> (format is Algorithm-Digest).
3	Decode and read the <code>certificateUrl</code> parameter from the notification.
4	Verify that the certificate corresponding to the URL is downloaded and cached.
5	If the certificate is not cached, download and cache it.
6	Follow steps 1 and 2 as specified in <a href="#">Working with Signatures</a> (p. 41) to create the <i>string-to-sign</i> . Include all the parameters in the notification except for the <code>signature</code> parameter.
7	Calculate the signature using the PKI based cryptography, using the <i>string-to-sign</i> you created in step 6 and the cached certificate in step 5..   <b>Note</b> Because the signature in <i>outbound notification</i> is calculated with the Amazon Payments private key using <i>PKI</i> , you need the values for the following parameters: <i>string-to-sign</i> , <code>signature</code> , <code>certificate</code> and <code>signatureMethod</code> . Each sample library uses these values in different formats. For information on using the sample library in the language of your choice see <a href="#">Getting the Samples</a> (p. 135)
8	Compare the calculated signature with the signature in the original notification.
9	If the signatures match, process the notification. Otherwise, discard the notification.



### Note

If you didn't choose the *Enable Signature V2* option from the [Developer and Seller Preferences](#) page, your responses will be signed using signature version 1. To verify signature version



1 signatures, you need to use the verification process described in [Appendix: Verifying Responses Signed Using Signature Version 1](#) (p. 137). For information on selecting signature version 2, see [About Signature Version 2](#) (p. 43)

## Access Key Rotation

If you decide that it is necessary to change your access keys, the security credentials page (available from your account page at the Amazon Web Services web site at <http://aws.amazon.com>) enables you to create a second set, and allows you to activate and deactivate the sets independently, as shown in the figure.

Created	Access Key ID	Secret Access Key	Status
Jun 12, 2009 07ZR7T8Y50859BSRG882	<a href="#">Show</a>		Active ( <a href="#">Make Inactive</a> )
Jun 10, 2009 0859BS07ZR7T8Y5RG882	<a href="#">Show</a>		Active ( <a href="#">Make Inactive</a> )

With both sets active, you can propagate the new set to your applications over time, maintaining the high security that signing provides. Since both sets are valid, you don't have to take your entire application down to incorporate the new keys. When the distribution is complete you can deactivate the old set.



### Note

You can have two sets of keys only. Both, one, or neither of them can be active.

## Soft Descriptor Customization

Credit card companies allow a descriptive string on credit card statements that identify a purchase. For example, `AMZN PMTS` appears on credit card statements to identify purchases made using Amazon FPS. Typically, most banks support a 19 character string. To give you more flexibility to identify yourself on credit card statements, Soft Descriptor Customization lets you modify information sent to the payment processor. You can use Soft Descriptor Customization in the following ways.

- Create a static string in your account settings



### Note

In marketplace applications the soft descriptor of the **recipient's** account settings are used when the soft descriptor type is *Static*.

- Create a dynamic string when you process the payment
- Specify whether the recipient or caller customer service number is sent to the payment processor

### How the Soft Descriptor Works

Option 1	You specify the soft descriptor string in your account level settings. Amazon FPS passes <code>AMZ*</code> plus the soft descriptor (in upper case) to the payment processor. For example, <code>AMZ*DIGITALDOWNLOAD</code> appears on the statement.
Option 2	You supply a sender description when processing the payment and specify <i>Dynamic</i> as the soft descriptor type. Amazon FPS passes a 19 character string, which consists of <code>AMZ*</code> plus the first 15 characters of the sender description (in upper case) to the payment processor. For example, if <code>AMAZON FPS GIFT</code> is the sender description then <code>AMZ*AMAZON FPS GIFT</code> appears on the statement.

Option 3 (default)	The default Amazon FPS descriptor <code>AMZN PMTS</code> appears on the statement if you do not specify a soft descriptor string in your account level settings or a sender description in your call to Amazon FPS.
--------------------	---

## SoftDescriptorType

Use the *SoftDescriptorType* in the [DescriptorPolicy](#) (p. 102) to specify static or dynamic soft descriptors. When you make a call to [Pay](#) (p. 74), FPS checks the *SoftDescriptorType* parameter in the [DescriptorPolicy](#) (p. 102). If you specify the parameter as *Static*, or do not specify a type, the soft descriptor in your account level setting is sent to the payment processor.



### Note

In *marketplace* applications, the soft descriptor of the **recipient's** account settings are used when the soft descriptor type is *Static*.

If you need a dynamic soft descriptor string, you must specify a sender description in the [Pay](#) (p. 74) action. You must also specify *Dynamic* as the soft descriptor type. Following the soft descriptor standard, the FPS soft descriptor consists of 19 characters beginning with the string `AMZ*`, followed by the first 15 characters of the sender description. You can use numbers, letters, or spaces in your soft descriptor as long as the descriptor doesn't begin or end with a space.

Special characters are not allowed in the soft descriptor string. Amazon FPS returns an error if you don't include a sender description for the dynamic string.

### To create a static soft descriptor

1. Log in to your Amazon Payments account at <http://payments.amazon.com>.
2. Point to **Edit My Account Settings**.
3. Click **Change My Business Settings**.
4. Enter the soft descriptor in the text box.

You can use numbers, letters, or spaces in your soft descriptor as long as the descriptor doesn't begin or end with a space. Special characters are not allowed in the soft descriptor string.

## CSOwner

In scenarios like marketplace applications, the caller and recipient are different parties. You can specify the customer service number that a customer sees on his credit card statement with the *CSOwner* parameter. When you make a call to the [Pay](#) (p. 74) action, FPS checks the *CSOwner* parameter. If you specify the value of the parameter as *Recipient*, or do not specify any value, the recipient's customer service number is determined from account information and sent to the payment processor. If you specify *Caller* as the value of the *CSOwner* parameter, the caller's customer service number is determined from account information and sent to the payment processor.



### Note

The soft descriptor and owner are passed to a [Reserve](#) (p. 85) operation are passed to the corresponding [Settle](#) (p. 90) operation.

The original soft descriptor and owner passed to the [Pay](#) (p. 74) or [Reserve](#) (p. 85) operations are passed to a corresponding [Refund](#) (p. 81) operation.

## Setting Up Instant Payment Notification

When the sender uses ABT (Amazon Payments Balance Transfer) to pay for a purchase, the purchase is approved or denied synchronously, which means that processing stops until the `Pay` call returns, and this happens relatively quickly. When the sender uses ACH (bank account withdrawal) or a credit card, the purchase is asynchronous, which means that it can take much longer to succeed or fail. Because you cannot know when asynchronous transactions will complete, Amazon FPS has created a notification service called Instant Payment Notification (IPN) that uses HTTP POST to notify you when the following asynchronous transactions occur:

- A payment or reserve succeeds
- A payment or reserve fails
- A payment or reserve goes into a pending state
- A reserved payment is settled successfully
- A reserved payment is not settled successfully
- A refund succeeds
- A refund fails
- A refund goes into a pending state
- A payment is canceled
- A reserve is canceled
- A token is canceled successfully
- A refund succeeds
- A refund fails
- A token is canceled successfully



### Note

IPN must be configured in order to operate. If you do not configure IPN, only email notifications will be sent.

IPN is a simple way to process updates from Amazon FPS and has the following benefits compared to other notification mechanisms:

- Easy implementation (compared to polling for updates)
- Robust delivery mechanism
- Robust to changes in message parameters
- Simple message structure



### Tip

If you have signed up for IPN and do not receive notifications, verify the URL you provided in your account settings. IPN will try for a day to deliver a notification before it gives up.

## Setting Up IPN Preferences

To receive IPN notifications, you need to set up a web service that receives IPN notifications from Amazon FPS and register the URL of that web service in your Amazon FPS developer account on <http://payments.amazon.com>.

If you decide to use IPN, you must sign in to your **Amazon Payments** account, and use the following procedure to enter the URL for your web server. Once you sign up for IPN, notifications are sent to your server.

### To configure your developer account so that you receive IPN messages

1. Log in to the Amazon Payments web site at <http://payments.amazon.com>.
2. Click **Edit My Account Settings**.  
The **Edit My Account Settings** page displays.
3. Click **Manage Developer and Seller Preferences**.  
The **Manage Developer and Seller Preferences** page displays.
4. Enter the URL for your IPN server in the **URL for Instant Payment Notification** text box.

## Receiving IPN Notifications

Amazon FPS uses HTTP POST to send IPN notifications to the URL registered in your Amazon Payments developer account. Use the following process to create a script that handles IPN notifications.



### Tip

If your IPN receiving service is down for some time, it is possible that our retry mechanism will deliver the IPNs out of order. If you receive an IPN for [TransactionStatus \(IPN\)](#) (p. 100), as SUCCESS or FAILURE or RESERVED, then after that time ignore any IPN that gives the PENDING status for the transaction.

### Setup Process for a Script to Receive IPN

1	Set up your web server to receive the HTTP POST IPN notifications on one of the following ports: 8080, 80 [http], 8443, or 443 [https].
2	Write a program that parses the IPN elements (for a list of the elements, see <a href="#">Common IPN Response Elements</a> (p. 49)).
3	Write your program so that it verifies the <i>signature</i> value sent in the IPN to make sure Amazon FPS sent the IPN. For more information, see <a href="#">Verifying the ReturnURL and IPN Notifications</a> (p. 43), below.
4	Write your program to use the returned elements to notify you of the IPN-related transactions.



### Important

The signature parameter *won't* be sent if you are using the SOAP protocol to call FPS API actions. We recommend that you set up an HTTPS endpoint using a standard Certificate Authority to receive IPN if you are using SOAP. Amazon FPS currently supports all the SUN JDK 1.5 CAs (cacerts file). In addition, we also support the standard CAs listed on <http://www.mozilla.org/projects/security/certs-included/>.

## How To Verify the IPN Signature

You must ensure that the IPN indeed came from Amazon Payments. You can do this by verifying the value of the *signature* parameter contained in the response. IPN responses contain the components you need to validate with either [client-side signature verification](#) or [client-side signature verification](#). For more information, see [Verifying the ReturnURL and IPN Notifications](#) (p. 43).

You can use the `IPNAndReturnURLValidation` sample to assist creating your own IPN validation page. For more information, see [Understanding the IPNAndReturnURLValidation Sample \(p. 133\)](#)

## Common IPN Response Elements

These response elements are common to all button transactions.

Name	Description
<code>addressFullName</code>	Full name of the buyer/sender. Type: String
<code>addressLine1</code>	Sender's address (first line). For IPN, this element is returned only if the value has been updated with Amazon. Type: String
<code>addressLine2</code>	Sender's address (second line). For IPN, this element is returned only if the value has been updated with Amazon. Type: String
<code>addressState</code>	Sender's state. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
<code>addressZip</code>	Sender's post code. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
<code>addressCountry</code>	Sender's country. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
<code>addressPhone</code>	Sender's phone number. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
<code>buyerEmail</code>	<p>Sender's e-mail address.</p> <p> <b>Note</b></p> <p>The <code>buyerEmail</code> element is not returned when the recipient is not the caller (i.e., marketplace transactions).</p> <p>Type: String</p>
<code>buyerName</code>	Sender's name. Type: String
<code>customData</code>	Data passed by the customer in the Pay call is returned in this element. Type: String
<code>customerEmail</code>	Customer's e-mail address. Type: String
<code>customerName</code>	Buyer/Sender Full Name. Type: String

**Amazon FPS Advanced Quick Start Developer Guide**  
**Receiving IPN Notifications**

Name	Description
<i>dateInstalled</i>	If the <i>notificationType</i> element (below) is <i>TokenCancellation</i> , this element contains the date the token was installed. Type: String
<i>integratorId</i>	If present, this is the id of the solution provider assisting with the transaction. Type: String
<i>isShippingAddressProvided</i>	If the IPN results include address updates, this element contains TRUE. Otherwise this element is not present in the response. Type: String
<i>operation</i>	The payment operation for this transaction. Type: String
<i>notificationType</i>	Notification type may be either <i>TokenCancellation</i> or <i>TransactionStatus</i> Type: String
<i>paymentMethod</i>	The payment method used by the sender. For more information, see the IPN values in <a href="#">PaymentMethod (p. 98)</a> . Type: String
<i>paymentReason</i>	Reason for payment. Type: String
<i>recipientEmail</i>	Recipient's e-mail address. Type: String
<i>recipientName</i>	Recipient's name. Type: String
<i>signature</i>	The encoded string the caller uses to verify the IPN. Amazon Payments calculates the signature using the elements in the returnURL. The merchant must have manually signed the request. For more information, see <a href="#">Handling the Receipt of IPN Notifications (p. 48)</a> . We recommend that you always verify the signature using the method in <a href="#">How to Verify the IPN Signature (p. 48)</a> . Type: String
<i>status</i>	Shorthand code that specifies the status of the transaction.. For more information, see <a href="#">TransactionStatus (IPN) (p. 100)</a> Type: String
<i>tokenId</i>	If <i>notificationType</i> is <i>TokenCancellation</i> , this element contains the ID of the cancelled token. Type: String
<i>tokenType</i>	If <i>notificationType</i> is <i>TokenCancellation</i> , this element contains the type of the cancelled token. Type: String

Name	Description
<i>transactionAmount</i>	Specifies the amount payable in this transaction; for example, USD 10.00. Type: String
<i>transactionDate</i>	The date when this transaction occurred, specified in seconds since the start of the epoch. Type: Long
<i>transactionId</i>	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. Type: String

## IPN Responses for Marketplace Transactions

The following IPN response elements are returned for marketplace transactions.

### IPN Marketplace Transaction Elements

Name	Description
<i>buyerName</i>	Sender's name. Type: String
<i>operation</i>	The payment operation for this transaction. Type: String
<i>paymentMethod</i>	The payment method used by the sender. For more information, see the IPN values in <a href="#">PaymentMethod (p. 98)</a> . Type: String
<i>paymentReason</i>	Reason for payment. Type: String
<i>recipientEmail</i>	Recipient's e-mail address. Type: String
<i>recipientName</i>	Recipient's name. Type: String
<i>referenceId</i>	If you specified a <code>referenceId</code> in the button creation form, Amazon Payments returns the <code>referenceId</code> to you. Type: String
<i>signature</i>	The encoded string the caller uses to verify the IPN. Amazon Payments calculates the signature using the elements in the <code>returnURL</code> . The merchant must have manually signed the request. For more information, see <a href="#">Handling the Receipt of IPN Notifications (p. 48)</a> . We recommend that you always verify the signature using the method in <a href="#">How to Verify the IPN Signature (p. 48)</a> . Type: String
<i>status</i>	Shorthand code that specifies the status of the transaction. For more information, see <a href="#">TransactionStatus (IPN) (p. 100)</a> . Type: String

**Amazon FPS Advanced Quick Start Developer Guide**  
**Receiving IPN Notifications**

---

<b>Name</b>	<b>Description</b>
<i>transactionAmount</i>	Specifies the amount payable in this transaction; for example, USD 10.00. This element is not being returned in the current version. Type: Double
<i>transactionDate</i>	The date when this transaction occurred, specified in seconds since the beginning of the epoch. Type: Long
<i>transactionId</i>	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. Type: String



# Amazon FPS API Reference

---

## Topics

- [Common Request Parameters \(p. 53\)](#)
- [Common Response Elements \(p. 54\)](#)
- [Error Codes \(p. 55\)](#)
- [Actions \(p. 62\)](#)
- [Data Types \(p. 96\)](#)

This section provides reference material for the Amazon FPS API.

The current version of the Amazon FPS API is 2008-09-17.

The WSDL is located at <https://fps.amazonaws.com/doc/2008-09-17/AmazonFPS.wsdl>.

The schema is located at <https://fps.amazonaws.com/doc/2008-09-17/AmazonFPS.xsd>.



## Note


To use the Amazon FPS API, you must have an Amazon FPS developer account. For information about getting the account, go to [Amazon Flexible Payments Service Getting Started Guide](#).

## Common Request Parameters

Each action in the API has its own specific set of parameters, but there is also a set of parameters that all actions use. This section describes those input parameters.

You only need to add these parameters in REST requests. SOAP requests include them by default.

The following table describes parameters that can be used in all Amazon FPS requests.

<b>Parameter</b>	<b>Description</b>	<b>Required</b>
<i>Action</i>	The API operation, for example, <code>Settle</code> or <code>Refund</code> . Type: String Default: None Constraint: Must be a valid operation such as <code>Cancel</code> , <code>Refund</code> , and so on.	Yes
<i>AWSSessionToken</i>	A string, distributed by Amazon FPS when you sign up to be a developer, that uniquely identifies the caller. Type: String Default: None	Yes
<i>Signature</i>	A value calculated using the request parameters and a SHA256 (preferred) or SHA1 HMAC encryption algorithm. Type: String Default: None	Yes
<i>SignatureVersion</i>	A value that specifies the <i>Signature</i> format. Type: Integer Default: None Valid Values: 1   2   <b>Important</b>  If you are currently using signature version 1 and are ready to migrate to signature version 2, please see <a href="#">Appendix: Moving your Application to Signature Version 2 (p. 139)</a> .	Yes
<i>SignatureMethod</i>	A value that specifies the signing method. Type: String Default: None Valid Values: <code>HmacSHA256</code> (preferred) and <code>HmacSHA1</code> .	Yes
<i>Timestamp</i>	A date-time value that marks the day and time the request was sent. Requests expire after a certain length of time to prevent malicious users from capturing requests and resubmitting them at a later time. Type: <code>dateTime</code> , for example, <code>2008-09-18T13:00:01Z</code> Default: None	Yes
<i>Version</i>	The version number of the WSDL to use in processing the request. Version numbers are dates, such as <code>2008-09-17</code> . For a list of version numbers, go to the Amazon Resource Center at <a href="http://aws.amazon.com/resources">http://aws.amazon.com/resources</a> . Type: String Default: None	Yes

## Common Response Elements

Each action in the API has its own set of response elements it uses. There are, however, a set of response elements that all actions use. The following table describes those common elements.

Element	Description
ResponseMetadata	Container element.
RequestId	Amazon FPS returns a <i>RequestId</i> element for every API call accepted for processing. The request ID is a reference to your API request that Amazon FPS can use to troubleshoot any issues related to the request. We recommend you store the request ID value for future reference. Because responses and requests can return asynchronously, you can use the request ID to sync responses with requests. Type: String

## Error Codes

Error	Description
<i>AccessFailure</i>	Account cannot be accessed.  You can display the following message to your customers:  Your account cannot be accessed. Retriable: Yes
<i>AccountClosed</i>	Account is not active.  You can display the following message to your customers: <i>Your account is closed.</i> Retriable: Yes
<i>AccountLimitsExceeded</i>	The spending or the receiving limit on the account is exceeded.  You can display the following message to your customers: <i>You have exceeded your spending or receiving limits. Please visit <a href="http://payments.amazon.com">http://payments.amazon.com</a> to update your payment limits.</i> Retriable: Yes
<i>AmountOutOfRange</i>	The transaction amount is more than the allowed range.  Ensure that you pass an amount within the allowed range. The transaction amount in a Pay operation using credit card or bank account must be greater than \$0.01. Retriable: No
<i>AuthFailure</i>	AWS was not able to validate the provided access credentials.  Please make sure that your AWS developer account is signed up for FPS. Retriable: Yes

**Amazon FPS Advanced Quick Start Developer Guide**  
**Error Codes**

---

<b>Error</b>	<b>Description</b>
<i>ConcurrentModification</i>	<p>A retrievable error can happen when two processes try to modify the same data at the same time.</p> <p>The developer should retry the request if this error is encountered. Retriable: Yes</p>
<i>DuplicateRequest</i>	<p>A different request associated with this caller reference already exists.</p> <p>You have used the same caller reference in an earlier request. Ensure that you use unique caller references for every new request.</p> <p>Even if your earlier request resulted in an error, you should still use a unique caller reference with every request and avoid this error. Retriable: No</p>
<i>InactiveInstrument</i>	<p>Payment instrument is inactive.</p> <p>The payment instrument is inactive, for example, a credit card has expired. Retriable: No</p>
<i>IncompatibleTokens</i>	<p>The transaction could not be completed because the tokens have incompatible payment instructions. If any assertion in one of the payment instructions fails, this error is displayed. As such, it may be caused by a number of reasons, for example:</p> <ul style="list-style-type: none"><li>• One or more tokens has expired.</li><li>• The recipient specified in the token is different from the actual recipient in the transaction.</li><li>• There is violation on the amount restriction.</li><li>• This token cannot be used with your application as another application has installed it.</li></ul>
<i>InstrumentAccessDenied</i>	<p>The external calling application is not the recipient for this postpaid or prepaid instrument. The caller should be the liability holder.</p> <p>You are trying to access an instrument that you do not own.</p>
<i>InstrumentExpired</i>	<p>The prepaid or the postpaid instrument has expired. You must ask your customers to set up a new prepaid or postpaid agreement</p>

**Amazon FPS Advanced Quick Start Developer Guide**  
**Error Codes**

<b>Error</b>	<b>Description</b>
<i>InsufficientBalance</i>	<p>The sender, caller, or recipient's account balance has insufficient funds to complete the transaction.</p> <p>You must ask your customers to fund their accounts. You can then retry this request.</p> <p>Funding an account can take up to three to four business days using a bank account transfer. This error is also displayed if the party paying the FPS fees does not have a sufficient account balance.</p> <p>Retriable: Yes</p>
<i>InternalError</i>	<p>A retrieable error that happens due to some transient problem in the system.</p> <p>The caller should retry the API call if this error is encountered.</p> <p>Retriable: Yes</p>
<i>InvalidAccountState_Caller</i>	<p>The developer account cannot participate in the transaction.</p> <p>Your account is not active. Contact your AWS Representative for more information.</p> <p>Retriable: Yes</p>
<i>InvalidAccountState_Recipient</i>	<p>Recipient account cannot participate in the transaction.</p> <p>You can display the following message to your customer (sender): <i>Your Amazon Payments account is not active. Please visit <a href="http://payments.amazon.com">http:// payments.amazon.com</a> for more details.</i></p> <p>Retriable: Yes</p>
<i>InvalidAccountState_Sender</i>	<p>Sender account cannot participate in the transaction.</p> <p>You can display the following message to your customer (sender): <i>Your Amazon Payments account is not active. Please visit <a href="http://payments.amazon.com">http://payments.amazon.com</a> for more details.</i></p> <p>Retriable: Yes</p>
<i>InvalidCallerReference</i>	<p>The Caller Reference does not have a token associated with it.</p> <p>Use the caller reference value that was passed to the InstallPaymentInstruction operation or the Amazon FPS Co-Branded UI pipeline.</p>
<i>InvalidClientTokenId</i>	<p>The AWS Access Key Id you provided does not exist in our records.</p> <p>Please check that the AWS Access Key Id used to make the request is valid.</p> <p>Retriable: No</p>

**Amazon FPS Advanced Quick Start Developer Guide**  
**Error Codes**

<b>Error</b>	<b>Description</b>
<i>InvalidDateRange</i>	The end date specified is before the start date or the start date is in the future. Specify the correct end date.
<i>InvalidParams</i>	One or more parameters in the request is invalid.  For more information, see the parameter descriptions for the action in the API Reference. Parameters are case sensitive. Retriable: No
<i>InvalidPaymentInstrument</i>	The payment method used in the transaction is invalid. Specify a valid payment method
<i>InvalidPaymentMethod</i>	<ul style="list-style-type: none"> <li>• For InstallPaymentInstruction, payment method specified in the GK construct is invalid. Specify the correct payment method.</li> <li>• For FundPrepaid and SettleDebt, the payment method specified in the token is invalid Use a token with payment method specified as any of ABT, ACH, and CC. For Quick accounts, only CC is acceptable.</li> </ul>
<i>InvalidRecipientForCCTransaction</i>	This account cannot receive credit card payments. You can display the following message to your customers: You cannot receive credit card payment. Please visit <a href="http://payments.amazon.com">http://payments.amazon.com</a> to update your account to receive credit card payments."
<i>InvalidSenderRoleForAccountType</i>	This token cannot be used for this operation.  Ensure that the account used in this transaction is the same account used in the original transaction. In a refund transaction, the recipient making the refund payment must be the same recipient as in the original transaction. Retriable: No
<i>InvalidTokenId</i>	You did not install the token that you are trying to cancel.  You do not have permission to cancel this token. You can cancel only the tokens that you own.  Retriable: No
<i>InvalidTokenId_Recipient</i>	The recipient token specified is either invalid or canceled.  You must install a new token if you are the recipient. If you are not the recipient, get a new payment authorization from the recipient.  Retriable: No

**Amazon FPS Advanced Quick Start Developer Guide**  
**Error Codes**

<b>Error</b>	<b>Description</b>
<i>InvalidTokenId_Sender</i>	<p>The send token specified is either invalid or canceled or the token is not active.</p> <p>You must ask your customer to set up a new payment authorization.</p> <p>Retriable: No</p>
<i>InvalidTokenType</i>	<p>An invalid operation was performed on the token, for example, getting the token usage information on a single use token.</p> <p>Retriable: No</p>
<i>InvalidTransactionId</i>	<p>The specified transaction could not be found or the caller did not execute the transaction or this is not a Pay or Reserve call.</p> <p>Specify the correct the transaction ID.</p> <p>Retriable: No</p>
<i>InvalidTransactionState</i>	<p>The transaction is not complete, or it has temporarily failed.</p> <p>Specify a duration of more than one hour.</p> <p>Retriable: No</p>
<i>NotMarketplaceApp</i>	<p>This is not an marketplace application or the caller does not match either the sender or the recipient.</p> <p>Please check that you are specifying the correct tokens.</p> <p>Retriable: Yes</p>
<i>OriginalTransactionFailed</i>	<p>The original transaction has failed.</p> <p>You cannot refund a transaction that has originally failed.</p> <p>Retriable: No</p>
<i>OriginalTransactionIncomplete</i>	<p>The original transaction is still in progress.</p> <p>Retry after the original transaction has completed.</p> <p>Retriable: Yes</p>
<i>PaymentInstrumentNotCC</i>	<p>The payment method specified in the transaction is not a credit card. You can only use a credit card for this transaction.</p> <p>Use only a credit card for this transaction.</p>
<i>PaymentMethodNotDefined</i>	<p>An attempt has been made to fund the prepaid instrument at a level greater than its recharge limit.</p> <p>Retriable: Yes, after adjusting the amount of funds to a valid level.</p>
<i>PrepaidFundingLimitExceeded</i>	<p>Payment method is not defined in the transaction.</p> <p>Specify the payment method in the sender token.</p>

**Amazon FPS Advanced Quick Start Developer Guide**  
**Error Codes**

---

<b>Error</b>	<b>Description</b>
<i>RefundAmountExceeded</i>	<p>The refund amount is more than the refundable amount.</p> <p>You are not allowed to refund more than the original transaction amount. Retriable: No</p>
<i>SameSenderAndRecipient</i>	<p>The sender and receiver are identical, which is not allowed. Retriable: No</p>
<i>SameTokenIdUsedMultipleTimes</i>	<p>This token is already used in earlier transactions. The tokens used in a transaction should be unique.</p>
<i>SenderNotOriginalRecipient</i>	<p>The sender in the refund transaction is not the recipient of the original transaction.</p> <p>The token you passed as the refund sender token does not belong to the recipient of the original transaction. Pass the correct refund sender token. Retriable: No</p>
<i>SettleAmountGreaterThanDebt</i>	<p>The amount being settled or written off is greater than the current debt.</p> <p>You cannot settle an amount greater than what is owed. Retriable: No</p>
<i>SettleAmountGreaterThanReserveAmount</i>	<p>The amount being settled is greater than the reserved amount.</p> <p>You cannot settle an amount greater than what is reserved. Retriable: No</p>
<i>SignatureDoesNotMatch</i>	<p>The request signature calculated by Amazon does not match the signature you provided.</p> <p>Check your AWS Secret Access Key and signing method. For more information, see "Working with Signatures" in the <a href="#">Amazon Flexible Payments Service Getting Started Guide</a>. Retriable: No</p>
<i>TokenAccessDenied</i>	<p>Permission is denied to cancel the token.</p> <p>You are not allowed to cancel this token. Retriable: No</p>



**Amazon FPS Advanced Quick Start Developer Guide**  
**Error Codes**

---

<b>Error</b>	<b>Description</b>
<i>TokenNotActive_Recipient</i>	<p>The recipient token is canceled.</p> <p>If you are the recipient, set up a new recipient token using the <code>InstallPaymentInstruction</code> operation or direct your customers to the Recipient Token Installation Pipeline to set up recipient token.</p> <p>Retriable: No</p>
<i>TokenNotActive_Sender</i>	<p>The sender token is canceled.</p> <p>You must ask your customer to set up a new payment authorization because the current authorization is not active.</p> <p>Retriable: No</p>
<i>TokenUsageError</i>	<p>The token usage limit is exceeded.</p> <p>If the usage has exceeded for this period, then wait for the next period before making another transaction. If the usage has exceeded for the entire authorization period, then ask your customer to set up a new payment authorization.</p>
<i>TransactionDenied</i>	<p>This transaction is not allowed.</p> <p>You are not allowed to do this transaction. Check your credentials.</p> <p>Retriable: No</p>
<i>TransactionFullyRefunded Already</i>	<p>This transaction has already been completely refunded.</p> <p>You are not allowed to refund more than the original transaction amount.</p> <p>Retriable: No</p>
<i>TransactionTypeNotRefundable</i>	<p>You cannot refund this transaction.</p> <p>Refund is allowed only on the <code>Pay</code> operation.</p> <p>Retriable: No</p>
<i>UnverifiedAccount_Recipient</i>	<p>The recipient's account must have a verified bank account or a credit card before this transaction can be initiated.</p> <p>You can display the following message to your customer (recipient): <i>Your Amazon Payments account is not active. Please visit <a href="http://payments.amazon.com">http://payments.amazon.com</a> for more details.</i></p> <p>Retriable: No</p>

<b>Error</b>	<b>Description</b>
<i>UnverifiedAccount_Sender</i>	<p>The sender's account must have a verified U.S. credit card or a verified U.S bank account before this transaction can be initiated.</p> <p>You can display the following message to your customers: <i>Please add a U.S. credit card or U.S. bank account and verify your bank account before making this payment.</i></p> <p>Retriable: No</p>
<i>UnverifiedBankAccount</i>	<p>A verified bank account should be used for this transaction.</p> <p>Visit the <a href="http://payments.amazon.com">http://payments.amazon.com</a> web site to verify your bank account.</p> <p>Retriable: No</p>
<i>UnverifiedEmailAddress_Caller</i>	<p>The caller account must have a verified e-mail address.</p> <p>You cannot make a web service API call without verifying your e-mail address. Go to <a href="http://payments.amazon.com">http://payments.amazon.com</a> web site and make payments.</p> <p>Retriable: No</p>
<i>UnverifiedEmailAddress_Recipient</i>	<p>The recipient account must have a verified e-mail address for receiving payments.</p> <p>You can display the following message to your customers: <i>You cannot receive payments. Please verify your e-mail address. Go to <a href="http://payments.amazon.com">http://payments.amazon.com</a> to verify your account and receive payments.</i></p> <p>Retriable: No</p>
<i>UnverifiedEmailAddress_Sender</i>	<p>The sender account must have a verified e-mail address for this payment</p> <p>You can display the following message to your customers: <i>You cannot receive payments. Please verify your e-mail address. Go to <a href="http://payments.amazon.com">http://payments.amazon.com</a> to verify your account and receive payments.</i></p> <p>Retriable: No</p>

## Actions

### Topics

- [Cancel \(p. 63\)](#)
- [CancelToken \(p. 65\)](#)
- [GetTokenByCaller \(p. 68\)](#)
- [GetTransactionStatus \(p. 71\)](#)
- [Pay \(p. 74\)](#)
- [Refund \(p. 81\)](#)

- [Reserve](#) (p. 85)
- [Settle](#) (p. 90)
- [VerifySignature](#) (p. 93)

This section describes the actions available with Amazon FPS Advanced Quick Start.

## Cancel

### Description

The `Cancel` action cancels a *reserved* or *pending* transaction. Once the transaction is canceled, you can't then settle it. You also can't use `Cancel` on a *completed* transaction. After a transaction is completed, you can do a refund if you want to reverse the order.

If the sender's credit card was in a reserved state, it is not part of this action to make sure the reserved status is removed. card.

### Request Parameters

Parameter	Description	Required
<i>Description</i>	Describes the reason for cancellation. Type: String Default: None	No
<i>TransactionId</i>	Specifies the transaction that needs to be canceled. This ID should have been returned by Amazon in a prior <code>Pay</code> or <code>Reserve</code> call. Type: String Default: None Constraint: Max size = 35 characters	Yes

For REST requests, you must also include parameters that are common to all requests. These parameters are included by default in SOAP requests. For more information, see [Common Request Parameters](#) (p. 53).

### Response Elements

Element	Description
<code>TransactionId</code>	The ID of the completed transaction. It is the same as the <code>TransactionID</code> provided in the request. Type: String
<code>TransactionStatus</code>	The status of the cancellation request. Type: <a href="#">TransactionStatus</a> (p. 100)

Responses also include elements common to all responses. For more information, see [Common Response Elements](#) (p. 54).

### Errors

This action can return the following errors:

- [AccessFailure](#) (p. 55)
- [AccountClosed](#) (p. 55)
- [AuthFailure](#) (p. 55)
- [ConcurrentModification](#) (p. 56)
- [InternalError](#) (p. 57)
- [InvalidClientTokenId](#) (p. 57)
- [InvalidParams](#) (p. 58)
- [InvalidTransactionState](#) (p. 59)
- [SignatureDoesNotMatch](#) (p. 60)

## Examples

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Cancel
&AWSAccessKeyId=AKIAIIFXJCFIHITREP4Q
&Description=MyWish
&Signature=yOedrTuiMoMrKt8SwugDDnfd0nydyoX9uPq1H1SUC14%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T09%3A14%3A58.796Z
&TransactionId=14GKI1SKSR1V6DO1RCCB32RBR6KLODMGQUD
&Version=2008-09-17
```

### Sample SOAP Request

```
https://fps.amazonaws.com/?
Action=Cancel
&AWSAccessKeyId=0656Example83G2
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2008-08-06T13%3A00%3A01Z
&TransactionId=254656Example83987
&Version=2008-09-17
&Signature=[URL-encoded signature value]
```

### Sample Response to REST Request

```
<CancelResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <CancelResult>
    <TransactionId>14GKI1SKSR1V6DO1RCCB32RBR6KLODMGQUD</TransactionId>
    <TransactionStatus>Cancelled</TransactionStatus>
  </CancelResult>
  <ResponseMetadata>
    <RequestId>6fe4b755-a328-419d-8967-e1d3b43779fc:0</RequestId>
  </ResponseMetadata>
</CancelResponse>
```

### Sample Response to SOAP Request

```
<CancelResponse
  xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <CancelResult>
```

```
<TransactionId>
  13N91G4R7478CJLIGROQH2VQJSM
</TransactionId>
<TransactionStatus>
  Success
</TransactionStatus>
</CancelResult>
</CancelResponse>
```

## Sample IPN Success Notification to Rest Request

```
-----
transactionId: 14GKI1SKSR1V6D01RCCB32RBR6KLODMGQUD
statusMessage: The transaction was explicitly cancelled by the caller.
transactionDate: 1254820475
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference08
transactionAmount: USD 1.00
transactionStatus: CANCELLED
operation: RESERVE
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature: jWDbBxtEhw2rQEyMeEXcpWCgoZvm8rjLEnmg38oYoPPR7NbMGgmMA9/5CDjt9Q/
FMktKMbARXnZF
YTzHj3YOKiAM3vxI0zT1oTiSdBx1KBRFzK7mauxxlQv5BYxjFX+R5cl+keCaT2nQyrp3agdrIIp5
MZ5Oy9dBuYMwMFWXoZZor90EidD23hBdZSOozQRUDzKaKJsF14RQVrKcf5pDCs1HaB6LBKbATaNT
RSxxrviIXy9JcWRQhJwzcc1H6cFOJDpNFSJ03b0Z94eL/XNu9BU7bt4KRwb+OHF0Pn53yf4zyBT9
jTD+94WeujCxE2rF0j5+brmXp/+Sn/RccDG7w==
recipientName: Test Business
paymentMethod: CC
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: Cancelled
-----
```

## CancelToken

### Description

The `CancelToken` action cancels a multi-use or recurring payment token . You can use this action at any time during the life of the token. After this request completes successfully, Amazon FPS stops all further payments that use the specified token. There is no way to reactivate a canceled token.

You can only cancel tokens that you created.

### Request Parameters

Parameter	Description	Required
<i>ReasonText</i>	Reason for canceling the payment token. Type: String Default: None	No

Parameter	Description	Required
<i>TokenId</i>	Specifies the token to cancel. You should have stored this value when it was returned as part of the response to the Co-Branded service request. Type: String Default: None	Yes

For REST requests, you must also include parameters that are common to all requests. These parameters are included by default in SOAP requests. For more information, see [Common Request Parameters](#) (p. 53).

## Response Elements

The response for this API includes only parameters common to all responses. For more information, see [Common Response Parameters](#) (p. 54).

## Errors

This action can return the following errors:

- [AccessFailure](#) (p. )
- [AccountClosed](#) (p. 55)
- [AuthFailure](#) (p. 55)
- [ConcurrentModification](#) (p. 56)
- [DuplicateRequest](#) (p. 56)
- [InternalError](#) (p. 57)
- [InvalidClientId](#) (p. 57)
- [InvalidParams](#) (p. 58)
- [InvalidTokenId](#) (p. 58)
- [SignatureDoesNotMatch](#) (p. 60)
- [TokenAccessDenied](#) (p. 60)

## Examples

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?  
Action=CancelToken  
&AWSAccessKeyId=AKIAIUQNNI2DNQHBO7RA  
&ReasonText=MyWish  
&Signature=IZD9O%2FWGqhkzO%2FdLTQ7Tn8KUAmtZXqIEg6gypwkGeWQ%3D  
&SignatureMethod=HmacSHA256  
&SignatureVersion=2  
&Timestamp=2009-10-07T08%3A46%3A37.156Z  
&TokenId=D739IT9TMC4FK9KB56PDKJWAQGXDZ3B8X3SJNGVH3UEF5GQ7XAQZMEIL4OGEZKGX  
&Version=2008-09-17
```

### Sample SOAP Request

```
<SOAP-ENV:Body  
wsu:id="body" xmlns:wsu=
```

```
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <ns2:CancelToken xmlns:ns2="http://fps.amazonaws.com/doc/2008-09-17/">
    <ns2:TokenId>
      B1HA64FUMCEB43QAJBTP3TVMJZGFLAX2DJJ3ZPAFHH1VNPGR74I83ZZI4HJ5NGEK
    </ns2:TokenId>
    <ns2:ReasonText>Buyer left the system.</ns2:ReasonText>
  </ns2:CancelToken>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Sample Response to REST Request

```
<CancelTokenResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <ResponseMetadata>
    <RequestId>a10e0ad6-148f-4afe-8bcd-e80a2680793d:0</RequestId>
  </ResponseMetadata>
</CancelTokenResponse>
```

## Sample Response to SOAP Request

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
      MESSAGE123
    </wsa:RelatesTo>
    <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
      CancelToken:Response
    </wsa:Action>
    <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing">
      urn:uuid:5fb6a949-a481-4f6f-bdfb-bcdde0daea6b
    </wsa:MessageID>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <CancelTokenResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
      <ResponseMetadata>
        <RequestId>5fb6a949-a481-4f6f-bdfb-bcdde0daea6b:0</RequestId>
      </ResponseMetadata>
    </CancelTokenResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Sample IPN Notification to Rest Request

```
-----
signatureVersion: 2
signatureMethod: RSA-SHA1
customerEmail: test-caller@amazon.com
tokenId: D739ATGTM94QK9NBU6P4KDWACGXZD8BVX3TJHGVP3XEFGME7XVQTMEL4OGFZMGP
callerReference: CallerReference19
notificationType: TokenCancellation
signature:
  flxZtuxk3jb0Ww4g4duMjx1s8EQnIC7kPHqKKu0t4trp1/8ZU6ohtm9V1xB1mdxDnJ37lpyfL7rp
  wE5tiKjJ8agm1OzPjp9rweVOEMcdscopTVhh9AG2HTNGyWyyaRlIPlXiV3mpPyMrttLiOkryB8ak
  YZ9fMbXUB9gKzMVzNhh58aayD/weMV/WIX3DDSJslsp0kg6frHv5F5CYrprwv4S+cXQxXdgJlRC3
```

```
UJO8bH68bwlFnyyzPz4+TnbB5xMDatpwkBOFCWO5+tmw1wJHyAUa7z6XJgwj27YIIjFSJolWlKwK
iZHqPNYNjKHE190sQMQLHcnkZeexig6wYHK5w==
tokenType: SingleUse
dateInstalled: Oct 8, 2009
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
customerName: Test Business
-----
```

## GetTokenByCaller

### Description

The `GetTokenByCaller` action returns the details about the token specified by a `tokenId` or `CallerReference`. The `CallerReference` is the value you passed in the Co-Branded service request, whereas the `tokenId` is the value you received in the Co-Branded service response.

### Request Parameters

Parameter	Description	Required
<code>CallerReference</code>	A value you provide that uniquely identifies the request. For more information, see <a href="#">Important Values to Store in Your Database (p. 30)</a> . Type: String Default: None Constraint: Max size = 128 characters Condition: Required if <code>tokenId</code> is not specified.	Conditional
<code>tokenId</code>	The sender token ID that the Co-Branded service returned. Type: String Default: None Constraint: Max size = 65 characters Condition: Required if <code>CallerReference</code> is not specified.	Conditional

For REST requests, you must also include parameters that are common to all requests. These parameters are included by default in SOAP requests. For more information, see [Common Request Parameters \(p. 53\)](#).

### Response Elements

Element	Description
Token	Details of the specified token. Type: <a href="#">Token (p. 104)</a>

Responses also include elements common to all responses. For more information, see [Common Response Elements \(p. 54\)](#).

### Errors

This action can return the following errors:



- [AccessFailure](#) (p. 55)
- [AccountClosed](#) (p. 55)
- [AuthFailure](#) (p. 55)
- [InternalError](#) (p. 57)
- [InvalidCallerReference](#) (p. 57)
- [InvalidClientTokenId](#) (p. 57)
- [InvalidParams](#) (p. 58)
- [InvalidTokenId](#) (p. 58)
- [SignatureDoesNotMatch](#) (p. 60)

## Examples

The following sections show a sample request and response.

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=GetTokenByCaller
&AWSAccessKeyId=AKIAIUQNNI2DNQHBO7RA
&CallerReference=callerReferenceSingleUse10
&Signature=7E43HRAge3s57KDtEW3%2Fv0CE3Rh4TkVuOpk%2FIU%2FJIEY%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-07T11%3A29%3A03.281Z
&TokenId=543IJMECGZZ3J4K1F7BJ3TMNXFBQU9VXNT7RRCTNAJDJ8X36L1ZRKSUUPPIBTTIK
&Version=2008-09-17
```

### Sample SOAP Request

```
<SOAP-ENV:Body wsu:Id="body"
xmlns:wsu=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">
  <ns2:GetTokenByCaller xmlns:ns2="http://fps.amazonaws.com/
doc/2008-09-17/">
    <ns2:CallerReference>
      ReferenceString????AxaM12275863261891
    </ns2:CallerReference>
  </ns2:GetTokenByCaller>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Sample Response to REST Request

```
<GetTokenByCallerResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <GetTokenByCallerResult>
    <Token>
      <TokenId>
        543IJMECGZZ3J4K1F7BJ3TMNXFBQU9VXNT7RRCTNAJDJ8X36L1ZRKSUUPPIBTTIK
      </TokenId>
      <FriendlyName>Friendly1339359778</FriendlyName>
      <TokenStatus>Active</TokenStatus>
      <DateInstalled>2009-10-07T04:29:05.054-07:00</DateInstalled>
      <CallerReference>callerReferenceSingleUse10</CallerReference>
    </Token>
  </GetTokenByCallerResult>
</GetTokenByCallerResponse>
```

```
<TokenType>SingleUse</TokenType>
<OldTokenId>
  543IJMECGZZ3J4K1F7BJ3TMNXFBQU9VXNT7RRCTNAJDJ8X36L1ZRKSUUPPIBTTIK
</OldTokenId>
<PaymentReason>PaymentReason</PaymentReason>
</Token>
</GetTokenByCallerResult>
<ResponseMetadata>
  <RequestId>45b6c560-8aa9-463c-84be-80eeefb21034:0</RequestId>
</ResponseMetadata>
</GetTokenByCallerResponse>
```

## Sample Response to SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
      MESSAGE123
    </wsa:RelatesTo>
    <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
      GetTokenByCaller:Response
    </wsa:Action>
    <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing">
      urn:uuid:4802991b-76dd-4f7b-8bd7-a1428cfbb9f2
    </wsa:MessageID>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <GetTokenByCallerResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17">
      <GetTokenByCallerResult>
        <Token>
          <TokenId>
            M157V8ZXS4AF9C132FNT5HXNNX8ZK7K1C5IHVRE4VZJTCEBLP8X1CTBCSFTFHKCK
          </TokenId>
          <FriendlyName>
            FriendlyName-pHhq12275863261891
          </FriendlyName>
          <TokenStatus>Active</TokenStatus>
          <DateInstalled>2008-04-24T20:12:06.200-08:00</DateInstalled>
          <CallerReference>ReferenceStringAxaM12275863261891</CallerReference>
          <TokenType>Unrestricted</TokenType>
          <OldTokenId>
            M157V8ZXS4AF9C132FNT5HXNNX8ZK7K1C5IHVRE4VZJTCEBLP8X1CTBCSFTFHKCK
          </OldTokenId>
          <PaymentReason>Testing</PaymentReason>
        </Token>
      </GetTokenByCallerResult>
      <ResponseMetadata>
        <RequestId>4802991b-76dd-4f7b-8bd7-a1428cfbb9f2:0</RequestId>
      </ResponseMetadata>
    </GetTokenByCallerResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## GetTransactionStatus

### Description

The `GetTransactionStatus` action returns the status of the transaction specified by the `TransactionId`. You could use this action if you choose not to process Instant Payment Notifications (IPNs) that you receive from Amazon Payments (for more information, see [Setting Up Instant Payment Notification \(p. 47\)](#)).

### Request Parameters

Parameter	Definition	Required
<code>TransactionId</code>	The transaction's ID. Type: String Constraint: Max size = 35 characters Default: None	Yes

For REST requests, you must also include parameters that are common to all requests. These parameters are included by default in SOAP requests. For more information, see [Common Request Parameters \(p. 53\)](#).

### Response Elements

Element	Description
<code>CallerReference</code>	A value you provide that uniquely identifies the request. Type: String
<code>StatusCode</code>	Shorthand code that specifies the status of the transaction. Expands on the information in the <code>TransactionStatus</code> field. For example, if <code>TransactionStatus</code> is PENDING, this field might be <code>PendingVerification</code> , or <code>PendingNetworkResponse</code> . Type: String Valid Values: See <a href="#">Status Codes (p. 71)</a>
<code>StatusMessage</code>	A description of the transaction status. Type: String
<code>TransactionId</code>	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. Type: String
<code>TransactionStatus</code>	The status of the transaction. Provides a short code on the status of the transaction, for example "PENDING". Type: <a href="#">TransactionStatus (p. 100)</a>

Responses also include elements common to all responses. For more information, see [Common Response Elements \(p. 54\)](#).

### Status Codes

This action can return the following values for `StatusCode`.

<b>Status Code</b>	<b>Message</b>
Canceled	The transaction was explicitly canceled by the caller.
Expired	This reserved amount on the payment instrument was not settled within the timeout period OR The transaction could not be completed within the specified timeout.
PendingNetworkResponse	This transaction is awaiting a response from the backend payment processor OR (Message returned by backend payment processor)
PendingVerification	The transaction has been flagged for manual investigation
Success	The requested amount was reserved successfully against the given payment instrument OR The transaction was successful and the payment instrument was charged.
TransactionDenied	(Message returned by backend payment processor) OR The transaction was denied after investigation.

## Errors

This action can return the following errors:

- [AccessFailure](#) (p. )
- [AuthFailure](#) (p. 55)
- [InternalError](#) (p. 57)
- [InvalidClientId](#) (p. 57)
- [InvalidParams](#) (p. 58)
- [InvalidTransactionId](#) (p. 59)
- [SignatureDoesNotMatch](#) (p. 60)

## Examples

The following sections show a sample request and response.

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=GetTransactionStatus
&AWSAccessKeyId=AKIAIIFXJCFIHITREP4Q
&Signature=2l60qD6%2BDIfVEN7ZiHM0AcUKACZt0GYKftIryqkCb6g%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T09%3A12%3A06.921Z
&TransactionId=14GKE3B85HCMF1BTSH5C4PD2IHZL95RJ2LM
```

&Version=2008-09-17

## Sample SOAP Request

```
GET\n
fps.sandbox.amazonaws.com\n
Action=GetTransactionStatus
&AWSAccessKeyId=AKIAIIFXJCFIHITREP4Q
&Signature=2l60qD6%2BDIfVEN7ZiHM0AcUKACZt0GYKftIryqkCb6g%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T09%3A12%3A06.921Z
&TransactionId=14GKE3B85HCMF1BTSH5C4PD2IHZL95RJ2LM
&Version=2008-09-17
```

## Sample Response to REST Request

```
<GetTransactionStatusResponse xmlns="http://fps.amazonaws.com/
doc/2008-09-17/">
  <GetTransactionStatusResult>
    <TransactionId>14GKE3B85HCMF1BTSH5C4PD2IHZL95RJ2LM</TransactionId>
    <TransactionStatus>Success</TransactionStatus>
    <CallerReference>CallerReference07</CallerReference>
    <StatusCode>Success</StatusCode>
    <StatusMessage>The transaction was successful and the payment instrument
was charged.</StatusMessage>
  </GetTransactionStatusResult>
  <ResponseMetadata>
    <RequestId>13279842-6f84-41ef-ae36-c1ededaf278d:0</RequestId>
  </ResponseMetadata>
</GetTransactionStatusResponse>
```

## Sample Response to SOAP Request

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/">
  <SOAP-ENV:Header>
    <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
      MESSAGE123
    </wsa:RelatesTo>
    <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
      GetTransactionStatus:Response
    </wsa:Action>
    <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing">
      urn:uuid:10d09e74-ba0a-4b3b-9eea-f873e589f496
    </wsa:MessageID>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <GetTransactionStatusResponse
xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
      <GetTransactionStatusResult>
        <TransactionId>13N9ZL42F2SJLLIGH7RB6Q8IO8BTM62LGI3</TransactionId>
        <TransactionStatus>Success</TransactionStatus>
        <CallerReference>ReferenceString????6geW12275895867941</
CallerReference>
        <StatusCode>Success</StatusCode>
      </GetTransactionStatusResult>
    </GetTransactionStatusResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```

        <StatusMessage>message</StatusMessage>
    </GetTransactionStatusResult>
    <ResponseMetadata>
        <RequestId>10d09e74-ba0a-4b3b-9eea-f873e589f496:0</RequestId>
    </ResponseMetadata>
</GetTransactionStatusResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## Pay

### Topics

- [Description \(p. 74\)](#)
- [Request Parameters \(p. 74\)](#)
- [Response Elements \(p. 77\)](#)
- [Errors \(p. 78\)](#)
- [Examples \(p. 78\)](#)
- [Related Actions \(p. 81\)](#)

## Description

The `Pay` action initiates a transaction to move funds from a sender to a recipient. The `SenderTokenId`, obtained from a Co-Branded service request, specifies the payment instrument the sender chose to execute the transaction. If the payment method specified is Amazon account balance transfer (ABT), the transaction completes synchronously. If the payment method is a bank account (ACH) or a credit card (CC), the transaction completes asynchronously. For more information about synchronous and asynchronous transactions, see [Setting Up Instant Payment Notification \(p. 47\)](#).

The marketplace implementation of `Pay` includes the recipient token ID, which identifies the recipient. You get this in the response from a marketplace Co-Branded service request (which you make when the recipient signs up on your web site for your marketplace services). The recipient token ID returned identifies the recipient and is required when you later move money from the sender to the recipient.

The `Pay` parameters also specify the marketplace fee and who is charged (the caller or recipient). The marketplace fee is typically the fee you charge the recipient for the service of hosting the recipient's e-commerce store. The fee can be charged on a per-transaction basis and consist of a flat fee, a percentage of the transaction, or a combination of the two.



## Request Parameters

Parameter	Description	Required
<i>CallerDescription</i>	Description of this transaction for the caller. Type: String Default: None Constraint: Max size = 160 characters	No
<i>CallerReference</i>	A value you provide that uniquely identifies the request. For more information, see <a href="#">Important Values to Store in Your Database (p. 30)</a> . Type: String Default: None Constraint: Max size = 128 characters	Yes

**Amazon FPS Advanced Quick Start Developer Guide**  
**Pay**

---

Parameter	Description	Required
<i>ChargeFeeTo</i>	<p>Specifies the participant paying the Amazon FPS fee in the transaction. The participant can only be a recipient or a caller. The following rules apply for specifying this parameter.</p> <ul style="list-style-type: none"><li>• If you are playing the role of a recipient and a caller, then set the value of this parameter to <i>Recipient</i>.</li><li>• If you are playing the role of caller and facilitating the transaction between a sender and a recipient, where the recipient pays the fee, then the fee is collected from the funds that are received from the sender.</li><li>• If you (caller) are paying the fees, then the fee is collected from your account balance. Ensure that you have a sufficient account balance to cover for the fees. If your account has an insufficient account balance, Amazon FPS rejects the transaction.</li></ul> <p>Type: String Default: Recipient Valid values: <i>Recipient</i>   <i>Caller</i></p>	No
<i>DescriptorPolicy</i>	<p>Specifies the entity whose name and contact details would be displayed in the sender's credit card or bank account statement.</p> <p>Type: <a href="#">DescriptorPolicy</a> (p. 102) Default: None</p>	No

Parameter	Description	Required
<i>MarketplaceFixedFee</i>	<p>Specifies the fee charged by the marketplace developer as a fixed amount of the transaction. The <i>MarketplaceFixedFee</i> is a separate fee from the Amazon Payments fee, which is paid by the caller or recipient. You can express the fixed fee as an amount, such as 10 to mean \$10. If you charge a variable fee per transaction, use the <i>MarketplaceVariableFee</i> parameter.</p> <p>Type: <a href="#">Amount</a> (p. 101)</p> <p>Default: If both the <i>MarketplaceFixedFee</i> and the <i>MarketplaceVariableFee</i> are unspecified, then the corresponding maximum values, if any, from the recipient token are used.</p> <p> <b>Important</b></p> <p>The value for <i>MarketPlaceFixedFee</i> must be less than or equal to the amount specified for the recipient token. If not, an <i>InvalidParams</i> error is returned with the following messages: "The MarketPlaceFixedFee (<i>\$amount-specified</i>) specified is greater than the maximum fixed fee (<i>\$amount-agreed</i>) agreed by the recipient.</p>	No
<i>MarketplaceVariableFee</i>	<p>Specifies the fee charged by the marketplace developer as a percentage of the transaction. The <i>MarketplaceVariableFee</i> is a separate fee from the Amazon Payments fee and is paid by the recipient. You can express the variable fee as a decimal, such as 5 to mean 5%. If you charge a fixed amount per transaction, use the <i>MarketplaceFixedFee</i> parameter.</p> <p>Type: Decimal</p> <p>Default: None</p> <p> <b>Important</b></p> <p>The value for <i>MarketPlaceVariableFee</i> must be less than or equal to the amount specified for the recipient token. If not, an <i>InvalidParams</i> error is returned with the following messages: "The MarketPlaceVariableFee (<i>\$amount-specified</i>) specified is greater than the maximum variable fee (<i>\$amount-agreed</i>) agreed by the recipient.</p>	No



<b>Parameter</b>	<b>Description</b>	<b>Required</b>
<i>RecipientTokenId</i>	Specifies the recipient token used in the transaction. You obtain this value in response from the Co-Branded service Recipient Token API (for more information, see <a href="#">Recipient Token API (p. 112)</a> . Type: String Default: None Condition: Required for marketplace transactions	Conditional
<i>SenderDescription</i>	Description of this transaction for the sender. If you use dynamic soft descriptors, you must specify a value for the sender description. Type: String Default: None Constraint: Max size = 160 characters Condition: If you use dynamic soft descriptors, you must specify a value for the sender description.	Conditional
<i>SenderTokenId</i>	Specifies the sender token used in the transaction. You obtain this value from the response to the Co-Branded service request. Type: String Default: None	Yes
<i>TransactionAmount</i>	Transaction amount charged to the sender for the purchase of an item or service. To understand how to correctly specify the amount in a REST request, see the example request at the end of this topic. Type: <a href="#">Amount (p. 101)</a> Default: None	Yes
<i>TransactionTimeoutInMins</i>	Specifies the number of minutes before the request times out. Use this parameter to specify a timeout value that is acceptable for your business. If Amazon FPS cannot complete the transaction in the time allotted, the transaction is marked as failed and you receive an IPN notification (if you are using IPN). Type: Integer (number of minutes) Default: 10080 (seven days)	No

You must also include parameters that are common to all requests. The common parameters are defaulted in SOAP calls but must be explicitly added in REST calls. For more information, see [Common Request Parameters \(p. 53\)](#).

## Response Elements

<b>Element</b>	<b>Description</b>
TransactionId	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. If the transaction is a Refund request, this parameter will contain the id of the Refund transaction only. Type: String

Element	Description
TransactionStatus	Provides the status of the transaction. Use this to determine if the transaction has completed, failed, or has not completed yet. Type: <a href="#">TransactionStatus</a> (p. 100)

Responses also include elements common to all responses. For more information, see [Common Response Elements](#) (p. 54).

Pay careful attention to all of the response elements listed in the preceding table, especially the response status element which indicates success or failure for the `Pay` operation. Errors are returned only for REST. For SOAP, an error results in a SOAP fault. If the response status is failure, the `Errors` element includes an error code that identifies the source of the failure. If the response status is success, the elements listed in the preceding table are returned.

## Errors

This action can return the following errors:

- [AccessFailure](#) (p. )
- [AccountLimitsExceeded](#) (p. 55)
- [AmountOutOfRange](#) (p. 55)
- [AuthFailure](#) (p. 55)
- [BadRule](#)
- [DuplicateRequest](#) (p. 56)
- [IncompatibleTokens](#) (p. 56)
- [InsufficientBalance](#) (p. 57)
- [InternalServerError](#) (p. 57)
- [InvalidAccountState\\_Caller](#) (p. 57)
- [InvalidAccountState\\_Recipient](#) (p. 57)
- [InvalidAccountState\\_Sender](#) (p. 57)
- [InvalidClientTokenId](#) (p. 57)
- [InvalidParams](#) (p. 58)
- [InvalidTokenId\\_Recipient](#) (p. 58)
- [InvalidTokenId\\_Sender](#) (p. 59)
- [NotMarketplaceApp](#) (p. 59)
- [PaymentMethodNotDefined](#) (p. 59)
- [SameSenderAndRecipient](#) (p. 60)
- [SameTokenIdUsedMultipleTimes](#) (p. 60)
- [SignatureDoesNotMatch](#) (p. 60)
- [TokenNotActive\\_Recipient](#) (p. 61)
- [TokenNotActive\\_Sender](#) (p. 61)
- [TokenUsageError](#) (p. 61)
- [TransactionDenied](#) (p. 61)
- [UnverifiedAccount\\_Recipient](#) (p. 61)
- [UnverifiedAccount\\_Sender](#) (p. 62)
- [UnverifiedBankAccount](#) (p. 62)
- [UnverifiedEmailAddress\\_Caller](#) (p. 62)
- [UnverifiedEmailAddress\\_Recipient](#) (p. 62)
- [UnverifiedEmailAddress\\_Sender](#) (p. 62)

The `Pay` action can return

## Examples

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Pay
&AWSAccessKeyId=AKIAIIFXJCFIHITREP4Q
&CallerDescription=MyWish
&CallerReference=CallerReference02
&SenderTokenId=553ILMLCG6Z8J431H7BX3UMN3FFQU8VSNSTRNCTAASDJNX66LNZLKSZU3PI7TXIH
&Signature=0AgvXMwJmLxwdMaie7lMHZxc6384h%2FjBkiTserQFpBQ%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T05%3A49%3A52.843Z
```

```
&TransactionAmount.CurrencyCode=USD
&TransactionAmount.Value=1
&Version=2008-09-17
```

## Sample SOAP Request

```
<SOAP-ENV:Body wsu:Id="body"
  xmlns:wsu=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">
  <ns2:Pay xmlns:ns2="http://fps.amazonaws.com/doc/2008-09-17/">
    <ns2:SenderTokenId>76PSX31MM77T81ExampleQVDNQP5GFAK</ns2:SenderTokenId>
    <ns2:TransactionAmount>
      <ns2:CurrencyCode>USD</ns2:CurrencyCode>
      <ns2:Amount>1.1</ns2:Amount>
    </ns2:TransactionAmount>
    <ns2:CallerReference>
      ReferenceString????rpXe12275876325471
    </ns2:CallerReference>
    <ns2:CallerDescription>
      DescriptionString-????i86x12275876325471
    </ns2:CallerDescription>
    <ns2:SenderDescription>
      DescriptionString-????0m6112275876325471
    </ns2:SenderDescription>
  </ns2:Pay>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Sample Response to REST Request

```
<PayResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <PayResult>
    <TransactionId>14GK6BGKA7U6OU6SUTNLBI5SBBV9PGDJ6UL</TransactionId>
    <TransactionStatus>Pending</TransactionStatus>
  </PayResult>
  <ResponseMetadata>
    <RequestId>c21e7735-9c08-4cd8-99bf-535a848c79b4:0</RequestId>
  </ResponseMetadata>
</PayResponse>
```

## Sample Response to SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
      MESSAGE123
    </wsa:RelatesTo>
    <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
      Pay:Response
    </wsa:Action>
    <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing">
      urn:uuid:b415f09d-5924-4315-b31a-21c977c85c39
    </wsa:MessageID>
```

```
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <PayResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
    <PayResult>
      <TransactionId>13N8UPFET32I4I7FCF9T4ZKFETETINTK56Q</TransactionId>
      <TransactionStatus>Pending</TransactionStatus>
    </PayResult>
    <ResponseMetadata>
      <RequestId>b415f09d-5924-4315-b31a-21c977c85c39:0</RequestId>
    </ResponseMetadata>
  </PayResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Sample IPN Pending Notification to Rest Request

```
-----
transactionId: 14GK6BGKA7U6OU6SUTNLBI5SBBV9PGDJ6UL
statusMessage: The transaction is awaiting a response from the backend
  payment processor.
transactionDate: 1254808208
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference02
transactionAmount: USD 1.00
transactionStatus: PENDING
operation: PAY
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature: uhp7uicAvF/
wTpRg6U279KTGPU2QHt23WiwNIB43i4ni1AEZOmBCTa3tUhlugwxvIMSRASBhiG0u
rU122IAXbt1iXfYprM2VrS0W0/W23BpkxInuNeAQWku4W5/uuOJlgVqyXsmxdFqJM7KKOh3IuUdC
wSfvPooR2qDQ2r5H/HjcOHfWQZk+BknXlw+aYpBRTa/mTYVxI6yq39mRyYPyMmh8r+tIPdevfnV1
B7sRlJhXkZJh6rHJEi7CHq4oqbf8HZ38xaaqyggWy310SmM0uY3YcxNng0T0dbkgNAozMIQgfOsL
4yxIyVIZZJEKFPgT/OdebCZkR/raY1JeuBdY0g==
recipientName: Test Business
paymentMethod: CC
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: MyWish
statusCode: PendingNetworkResponse
-----
```

## Sample IPN Success Notification to Rest Request

```
-----
transactionId: 14GK6BGKA7U6OU6SUTNLBI5SBBV9PGDJ6UL
statusMessage: The transaction was successful and the payment instrument was
  charged.
transactionDate: 1254808208
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference02
transactionAmount: USD 1.00
transactionStatus: SUCCESS
operation: PAY
```

```

recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature: yuYUR4IkONbOfrrerafrzC6raA90suk
+jKXCgaV1LY0DxieYCAG2tAf9S7Rt231kzr0mhMMOIH0oe
ochId3zdXp+2VaUbE4qGjPGfImpaBVxtxVwcdQP6cSFnvnKAbPbmQMdeIHMLgDeqVdtu5BO5skwj
e6bkDs+b8TQ3pHBYmXdc69aHceGqWAjMujs6m4HH3Oth1b5Rj54s1IedwTi63HyQo+IAyRWvGPTn
nT6YlV0ajG38GCPoS9Wqa+UKcIr0sLoPY0y2StCDyjYHz7iVx+6lzGleeCmZ++rAKU8swwhBiWGZ
56ajlKTzhoIJnK5yk7jFYreRt+Ff0W2fEnvEyQ==
recipientName: Test Business
paymentMethod: CC
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: MyWish
statusCode: Success
-----

```

## Related Actions

- [Refund \(p. 81\)](#)

## Refund

### Description

You use `Refund` to refund a successfully completed payment transaction. You can refund less than the amount paid. The default, however, is to refund the full amount to the sender.

Only the caller of the original transaction can perform a refund.

### Request Parameters

Parameter	Description	Required
<i>CallerDescription</i>	Description of this transaction for the caller. Type: String Default: None Constraint: Max size = 160 characters	No
<i>CallerReference</i>	A value you provide that uniquely identifies the request. For more information, see <a href="#">Important Values to Store in Your Database (p. 30)</a> . Type: String Default: None Constraint: Max size = 128 characters	Yes
<i>RefundAmount</i>	Specifies the amount to be refunded. To understand how to correctly specify the amount in a REST request, see the example request at the end of this topic. Type: <a href="#">Amount (p. 101)</a> Default: Original transaction amount or any amount remaining Constraint: The total refund amount cannot exceed the original transaction amount.	No

Parameter	Description	Required
<i>TransactionId</i>	Transaction ID of the transaction to be refunded. The transaction ID must be one returned from a <code>Pay</code> action. Type: String Default: None Constraint: Max size = 35 characters	Yes

For REST requests, you must also include parameters that are common to all requests. These parameters are included by default in SOAP requests. For more information, see [Common Request Parameters](#) (p. 53).

## Response Elements

Element	Description
<code>TransactionId</code>	This is the ID (max size = 35 characters) of the transaction named in the request. Type: String
<code>TransactionStatus</code>	Provides the status of the transaction. Type: <a href="#">TransactionStatus</a> (p. 100)

Responses also include elements common to all responses. For more information, see [Common Response Elements](#) (p. 54).

## Errors

This action can return the following errors:

- [AccessFailure](#) (p. 55)
- [AmountOutOfRange](#) (p. 55)
- [AuthFailure](#) (p. 55)
- [ConcurrentModification](#) (p. 56)
- [DuplicateRequest](#) (p. 56)
- [InternalServerError](#) (p. 57)
- [InvalidAccountState\\_Caller](#) (p. 57)
- [InvalidAccountState\\_Recipient](#) (p. 57)
- [InvalidAccountState\\_Sender](#) (p. 57)
- [InvalidClientTokenId](#) (p. 57)
- [InvalidParams](#) (p. 58)
- [InvalidTransactionId](#) (p. 59)
- [OriginalTransactionFailed](#) (p. 59)
- [OriginalTransactionIncomplete](#) (p. 59)
- [RefundAmountExceeded](#) (p. 60)
- [SignatureDoesNotMatch](#) (p. 60)
- [TransactionDenied](#) (p. 61)
- [TransactionFullyRefundedAlready](#) (p. 61)
- [TransactionTypeNotRefundable](#) (p. 61)
- [UnverifiedEmailAddress\\_Caller](#) (p. 62)
- [UnverifiedEmailAddress\\_Sender](#) (p. 62)

## Examples

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Refund
&AWSAccessKeyId=AKIAIIFXJCFIHITREP4Q
&CallerDescription=MyWish
```

```
&CallerReference=CallerReference03
&RefundAmount.CurrencyCode=USD
&RefundAmount.Value=1
&Signature=V6pU3PvDPkPhR9Eu7yZXnFZHUEFafLE5sBPgqqCELEU%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T05%3A51%3A49.578Z
&TransactionId=14GK4TNCAQ84NK9VITEHKAS94RAD9ZE2AQD
&Version=2008-09-17
```

## Sample SOAP Request

```
<SOAP-ENV:Body wsu:Id="body"
  xmlns:wsu=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">
  <ns2:Pay xmlns:ns2="http://fps.amazonaws.com/doc/2008-09-17/">
    <ns2:SenderTokenId>76PSX31MM77T81ExampleQVDNQP5GFAK</ns2:SenderTokenId>
    <ns2:RecipientTokenId>6SC9UJ1VJEEExampleBTBNUNEYUBJM1K</
ns2:RecipientTokenId>
    <ns2:TransactionAmount>
      <ns2:CurrencyCode>USD</ns2:CurrencyCode>
      <ns2:Amount>1.1</ns2:Amount>
    </ns2:TransactionAmount>
    <ns2:CallerReference>
      ReferenceString????rpXe12275876325471
    </ns2:CallerReference>
    <ns2:CallerDescription>
      DescriptionString-????i86x12275876325471
    </ns2:CallerDescription>
    <ns2:SenderDescription>
      DescriptionString-????0m6112275876325471
    </ns2:SenderDescription>
  </ns2:Pay>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Sample Response to REST Request

```
<RefundResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <RefundResult>
    <TransactionId>14GK6F2QU755ODS27SGHEURLKPG72Z54KMF</TransactionId>
    <TransactionStatus>Pending</TransactionStatus>
  </RefundResult>
  <ResponseMetadata>
    <RequestId>1a146b9a-b37b-4f5f-bda6-012a5b9e45c3:0</RequestId>
  </ResponseMetadata>
</RefundResponse>
```

## Sample Response to SOAP Request

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
      MESSAGE123
    </wsa:RelatesTo>
    <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
```

```
<wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
  Refund:Response
</wsa:Action>
<wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing">
  urn:uuid:6d63adbb-611e-40ee-9262-a29c30e8ecaa
</wsa:MessageID>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <RefundResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
    <RefundResult>
      <TransactionId>13N91G4R7478C8ZLHEF93JLIGROQH2VQJSM</TransactionId>
      <TransactionStatus>Success</TransactionStatus>
    </RefundResult>
    <ResponseMetadata>
      <RequestId>6d63adbb-611e-40ee-9262-a29c30e8ecaa:0</RequestId>
    </ResponseMetadata>
  </RefundResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Sample IPN Pending Notification to Rest Request

```
-----
transactionId: 14GK6F2QU755ODS27SGHEURLKPG72Z54KMF
statusMessage: The transaction is awaiting a response from the backend
  payment processor.
transactionDate: 1254808324
signatureVersion: 2
signatureMethod: RSA-SHA1
parentTransactionId: 14GK4TNCAQ84NK9VITEHKAS94RAD9ZE2AQD
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference03
transactionAmount: USD 1.00
transactionStatus: PENDING
operation: REFUND
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature:
  mziS1HbeiiLx5j8nrUR3UeIVz3bcxVDG82JOW0gIEX01FXxBVZHwPPBFCEVcyBMu8wtNTMph/yLu
  okjBi8w9Q6shMswBteq9bwnQA9qbDRT256ckoqdwfCf09101YVj+wNSKkezF6Clptjgsn0wMjMQO
  D9QBuOAAA9qV6VnUorRumPZlpsY/17FUVdWkVUMPEkZNO1mn7lcLFZJJp1aMkIj+RmraafTUUM62
  U0VMYKSR5pDEp0ifThn0Za4DogV0ZoGJrB/+gPhA07FdtnkM4uG5jgwgqOCVyOA4ayP7uJpb7oImj
  8Jhi60+EWUUbbsUSHTEsjTxqQtM8UKvsM6XAjdA==
recipientName: Test Business
paymentMethod: CC
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: MyWish
statusCode: PendingNetworkResponse
-----
```

## Sample IPN Success Notification to Rest Request

```
-----
transactionId: 14GK6F2QU755ODS27SGHEURLKPG72Z54KMF
statusMessage: The transaction was successful and the payment instrument was
  charged.
transactionDate: 1254808324
```



```
signatureVersion: 2
signatureMethod: RSA-SHA1
parentTransactionId: 14GK4TNCAQ84NK9VITEHKAS94RAD9ZE2AQD
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference03
transactionAmount: USD 1.00
transactionStatus: SUCCESS
operation: REFUND
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature: sDq9YvW7L29W2NSIC/
wjC5yLyR4QJSQyt/7iHhNiEdwFoGVkrLjJHiBlopFjXzznHnmMtCRsUQ+A
d3tZ0NdemMxf0qYM9NX93PyG0KBKXShKeM0Da39cvnC05tZmtxpfCuZT5ECRydr+BqRo/D0lx1Yg
93gihZ83qHWR8bpqQcBwsu7vD4c4m4mTZ4I75gw+NXXRDD+vCPFDNEKRnh5kQz+Tjjg4bnNYEEcG
Rf6UZfS2lvMzdj0c37RUY6t4gQ3W3Z9G/REGjC98JBUtimk/kc1HoSc+xe6WtAH/siNurisyqgoB
HwnQM8iRqLEHj/m9y6vx5EBHBokD1BJMIiiZNg==
recipientName: Test Business
paymentMethod: CC
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: MyWish
statusCode: Success
-----
```

## Related Actions

- [Pay](#) (p. 74)

## Reserve

### Description

The `Reserve` operation reserves the total price of a purchase against the sender's payment instrument. To charge the payment instrument, you must subsequently issue a `Settle` request. A reserve authorization is only valid for 7 days. After that, Amazon FPS automatically cancels the transaction and notifies you.



#### Note

You can settle a reserved transaction only once.

To cancel a reserved payment, send a `Cancel` request.

### Request Parameters

Parameter	Description	Required
<code>CallerDescription</code>	Description of this transaction for the caller. Type: String Default: None Constraint: Max size = 160 characters Condition: If you use dynamic, soft descriptors, you must supply a caller description. For more information, see <i>DescriptorPolicy</i> .	No

Parameter	Description	Required
<i>CallerReference</i>	A value you provide that uniquely identifies the request. For more information, see <a href="#">Important Values to Store in Your Database (p. 30)</a> . Type: String Default: None Constraint: Max size = 128 characters	Yes
<i>DescriptorPolicy</i>	Specifies the entity whose name and contact details would be displayed in the sender's credit card or bank account statement. Type: <a href="#">Descriptor Policy (p. 102)</a> Default: None	No
<i>SenderDescription</i>	Description of this transaction for the sender. If you use dynamic soft descriptors, you must specify a value for the sender description. Type: String Default: None Constraint: Max size = 160 characters Condition: If you use dynamic soft descriptors, you must specify a value for the sender description. For more information, see <i>DescriptorPolicy</i> .	Conditional
<i>SenderTokenId</i>	Specifies the sender token to be used for this transaction. You obtain this value in a Co-Branded service response. Type: String Default: None	Yes
<i>TransactionAmount</i>	Transaction amount charged to the sender. To understand how to correctly specify the amount in a REST request, see the example request at the end of this topic. Type: <a href="#">Amount (p. 101)</a> Default: None	Yes

You must also include parameters that are common to all requests. The common parameters are defaulted in SOAP calls but must be explicitly added in REST calls. For more information, see [Common Request Parameters \(p. 53\)](#).

## Response Elements

Element	Description
<i>TransactionId</i>	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. If the transaction is a Refund request, this parameter will contain the id of the Refund transaction only. Type: String
<i>TransactionStatus</i>	Provides the status of the transaction. Type: <a href="#">TransactionStatus (p. 100)</a>

Responses also include elements common to all responses. For more information, see [Common Response Elements](#) (p. 54).

## Errors

This action can return the following errors:

- [AccessFailure](#) (p. 54)
- [AccountLimitsExceeded](#) (p. 55)
- [AmountOutOfRange](#) (p. 55)
- [AuthFailure](#) (p. 55)
- [DuplicateRequest](#) (p. 56)
- [IncompatibleTokens](#) (p. 56)
- [InternalError](#) (p. 57)
- [InvalidAccountState\\_Caller](#) (p. 57)
- [InvalidAccountState\\_Recipient](#) (p. 57)
- [InvalidAccountState\\_Sender](#) (p. 57)
- [InvalidClientTokenId](#) (p. 57)
- [InvalidParams](#) (p. 58)
- [InvalidPaymentMethod](#) (p. 58)
- [InvalidRecipientForCCTransaction](#) (p. 58)
- [InvalidTokenId\\_Sender](#) (p. 59)
- [PaymentInstrumentNotCC](#) (p. 59)
- [SignatureDoesNotMatch](#) (p. 60)
- [TokenNotActive\\_Recipient](#) (p. 61)
- [TokenNotActive\\_Sender](#) (p. 61)
- [TransactionDenied](#) (p. 61)
- [UnverifiedAccount\\_Recipient](#) (p. 61)
- [UnverifiedAccount\\_Sender](#) (p. 62)
- [UnverifiedEmailAddress\\_Caller](#) (p. 62)
- [UnverifiedEmailAddress\\_Recipient](#) (p. 62)
- [UnverifiedEmailAddress\\_Sender](#) (p. 62)

## Examples

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Reserve
&AWSAccessKeyId=AKIAIIFXJCFIHITREP4Q
&CallerDescription=Reserve
&CallerReference=CallerReference05
&SenderTokenId=553IPMACGAZ2J4N1L7BJ3UMNRFQTQU4V9NT4RJCTVADDJKXQ6L1ZAKSIUNPIRTTI1
&Signature=JZ0eeVTM5LwbvziLdA%2FSMve7mgrEoTvTGZJ%2BpsgZkM%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T07%3A51%3A04.140Z
&TransactionAmount.CurrencyCode=USD
&TransactionAmount.Value=1
&Version=2008-09-17
```

### Sample SOAP Request

```
<SOAP-ENV:Body wsu:Id="body" xmlns:wsu=
"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">
  <ns2:Reserve xmlns:ns2="http://fps.amazonaws.com/doc/2008-09-17/">
    <ns2:SenderTokenId>76PSX31MM77T81ExampleQVDNQP5GFAK</ns2:SenderTokenId>
    <ns2:TransactionAmount>
      <ns2:CurrencyCode>USD</ns2:CurrencyCode>
      <ns2:Amount>1.10</ns2:Amount>
    </ns2:TransactionAmount>
    <ns2:CallerReference>
      ReferenceString????Qlrd12275864150791
    </ns2:CallerReference>
  </ns2:Reserve>
</SOAP-ENV:Body>
```

```
</ns2:CallerReference>
<ns2:CallerDescription>
  DescriptionString-????UQAu12275864150791
</ns2:CallerDescription>
<ns2:SenderDescription>
  DescriptionString-????kbPT12275864150791
</ns2:SenderDescription>
</ns2:Reserve>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Sample Response to REST Request

```
<ReserveResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <ReserveResult>
    <TransactionId>14GKD9GE66FAA63E606B2JDPZKN53LZ7F22</TransactionId>
    <TransactionStatus>Pending</TransactionStatus>
  </ReserveResult>
  <ResponseMetadata>
    <RequestId>d13273fc-fca8-4963-8fbc-66d03e66055f:0</RequestId>
  </ResponseMetadata>
</ReserveResponse>
```

## Sample Response to SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
      MESSAGE123
    </wsa:RelatesTo>
    <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
      Reserve:Response
    </wsa:Action>
    <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing">
      urn:uuid:a9e1fc80-03f6-4e1b-alc0-541df545afac
    </wsa:MessageID>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ReserveResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
      <ReserveResult>
        <TransactionId>13N8TKAK15P3GOIPLP796OKSB66C6K2LBKE</TransactionId>
        <TransactionStatus>Pending</TransactionStatus>
      </ReserveResult>
      <ResponseMetadata>
        <RequestId>a9e1fc80-03f6-4e1b-alc0-541df545afac:0</RequestId>
      </ResponseMetadata>
    </ReserveResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Sample IPN Pending Notification to Rest Request

```
-----
transactionId: 14GKD9GE66FAA63E606B2JDPZKN53LZ7F22
```

```
statusMessage: The transaction is awaiting a response from the backend
  payment processor.
transactionDate: 1254815482
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference05
operation: RESERVE
transactionStatus: PENDING
transactionAmount: USD 1.00
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature: NvFCZMralNEepynuIhhXJc+jpK1ZMdFLBMcXFv6Vq1jhpDLX/
B9T0lluOUv74I6xgO8L2UemgV4S
ZCejlQZ3glwKnEM751KVlHx34IKp1RFm1DjQO05KaYGQUNMulouYK1YmQUHCuktdLnTXjxjnlv
9U4EyzDe8l/tLp2nlAqRF4J7PIhdTkWvBYNYhZrEy5A895OMf9uFtwX8Eyg4lTDMVwEWJoG8CTxJ
qtcsKabmbF9Blwhfe3f+viTnv39YRDb+PZKnpl/XqkKYdNEXClRy3g6xpF/14FJ4hA+A1UP+A+No
17b6lZuKmd5dbdvqTQKOxEaFR6lL1gtZAYY/8w==
recipientName: Test Business
paymentMethod: CC
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: PendingNetworkResponse
-----
```

## Sample IPN Success Notification to Rest Request

```
-----
transactionId: 14GKD9GE66FAA63E606B2JDPZKN53LZ7F22
statusMessage: The requested amount was reserved successfully against the
  given payment instrument.
transactionDate: 1254815482
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference05
transactionAmount: USD 1.00
transactionStatus: RESERVED
operation: RESERVE
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature: RIVZQHF+NmGUEbZNXijRcSwmeBTcYg/GCZD/
xeUpLLXmWdNrM1D0+ewFLiUqJvdbQueUilBkJPoB
5j+ZYvvrXfldEofaMZ85pz2pA/DyUicWR4e/DgcZrk/B7F06LL9ki6aE0qPzpRR/nzRcLiu1lH2a
zUPnMVf3dT+SfDhaKyKIxfX40QYL6U3m3NTaGYSUBwZczg9qTpu4zZ2kCK3uidg7P78sXQEnDhm
8kDAJC4obYFVlZi/Bd8UalxIYf2ko8SkhQ4vbsipjNg++HJ7KlJaa4lGTVCrJfeX0Y4r7ToONEaQ
iu/zn8X+q/jPqgGZN+Z2KNls6XVw4Waw3eXbug==
recipientName: Test Business
paymentMethod: CC
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: Success
-----
```

## Related Actions

- [Pay](#) (p. 74)
- [Refund](#) (p. 81)
- [Settle](#) (p. 90)

## Settle

### Description

The `Settle` action charges the sender's payment instrument for the purchase that was transacted using `Reserve`. You settle a transaction when you fulfill the order, for example, when you ship the purchased items.

### Request Parameters

Parameter	Description	Required
<code>ReserveTransactionId</code>	An identifier returned by <code>Reserve</code> that identifies the reserved transaction to be settled. Type: String Default: None Constraint: Max size = 35 characters	Yes
<code>TransactionAmount</code>	Amount to be settled. To understand how to correctly specify the amount in a REST request, see the example request at the end of this topic. Type: <a href="#">Amount</a> (p. 101) Default: The amount reserved in the <code>Reserve</code> request Constraint: The amount cannot exceed the reserved amount.	No

For REST requests, you must also include parameters that are common to all requests. These parameters are included by default in SOAP requests. For more information, see [Common Request Parameters](#) (p. 53).

### Response Elements

Element	Description
<code>TransactionId</code>	Identifies the transaction that was settled. Type: String
<code>TransactionStatus</code>	Provides the status of the transaction. Type: <a href="#">TransactionStatus</a> (p. 100)

Responses also include elements common to all responses. For more information, see [Common Response Elements \(p. 54\)](#).

## Errors

This action can return the following errors:

- [AccessFailure \(p. 55\)](#)
- [AccountClosed \(p. 55\)](#)
- [AmountOutOfRange \(p. 55\)](#)
- [AuthFailure \(p. 55\)](#)
- [ConcurrentModification \(p. 56\)](#)
- [InternalError \(p. 57\)](#)
- [InvalidAccountState\\_Caller \(p. 57\)](#)
- [InvalidAccountState\\_Recipient \(p. 57\)](#)
- [InvalidAccountState\\_Sender \(p. 57\)](#)
- [InvalidClientTokenId \(p. 57\)](#)
- [InvalidParams \(p. 58\)](#)
- [InvalidTransactionId \(p. 59\)](#)
- [InvalidTransactionState \(p. 59\)](#)
- [SettleAmountGreaterThanReserveAmount \(p. 60\)](#)
- [SignatureDoesNotMatch \(p. 60\)](#)
- [TransactionDenied \(p. 61\)](#)
- [UnverifiedAccount\\_Recipient \(p. 61\)](#)
- [UnverifiedEmailAddress\\_Caller \(p. 62\)](#)
- [UnverifiedEmailAddress\\_Recipient \(p. 62\)](#)
- [UnverifiedEmailAddress\\_Sender \(p. 62\)](#)

## Examples

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Settle
&AWSAccessKeyId=AKIAIIFXJCFIHITREP4Q
&ReserveTransactionId=14GKD9GE66FAA63E606B2JDPZKN53LZ7F22
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Signature=SJJLsIBghi7VIycBjX7c3hnfgZ%2FBvZbzqLtAZXDL8ys%3D
&Timestamp=2009-10-06T07%3A53%3A11.750Z
&TransactionAmount.CurrencyCode=USD
&TransactionAmount.Value=1
&Version=2008-09-17
```

### Sample SOAP Request

```
https://fps.amazonaws.com/?
Action=Settle
&AWSAccessKeyId=0656Example83G2
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2008-08-06T13%3A00%3A01Z
&TransactionId=254656Example83987
&Version=2008-09-17
&Signature=<URL-encoded signature value>
```

### Sample Response to REST Request

```
<SettleResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <SettleResult>
    <TransactionId>14GKD9GE66FAA63E606B2JDPZKN53LZ7F22</TransactionId>
    <TransactionStatus>Pending</TransactionStatus>
```

```
</SettleResult>
<ResponseMetadata>
  <RequestId>9ed2008b-b230-4ed0-9210-095f77fc2359:0</RequestId>
</ResponseMetadata>
</SettleResponse>
```

## Sample Response to SOAP Request

```
<SettleResponse
  xmlns="https://fps.amazonaws.com/doc/2008-09-17/">
  <SettleResult>
    <TransactionId>
      254656Example83987
    </TransactionId>
    <TransactionStatus>
      Pending
    </TransactionStatus>
  </SettleResult>
  <ResponseMetadata>
    <RequestId>
      a8d5e97c-6a7e-4fe1-b019-58a428a5a68b:0
    </RequestId>
  </ResponseMetadata>
</SettleResponse>
```

## Sample IPN Pending Notification to Rest Request

```
-----
transactionId: 14GKD9GE66FAA63E606B2JDPZKN53LZ7F22
statusMessage: The transaction is awaiting a response from the backend
  payment processor.
transactionDate: 1254815482
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference05
transactionAmount: USD 1.00
transactionStatus: PENDING
operation: SETTLE
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature: zxyWmlhu4o
+2rpdRbXU08EACZ3Mi3Z16x5+8+1Hbqkh4DTr1A6ry4fijBYkl32z4fMF9xnoGriW
2jzij7Vmc/4Vc4dEWCpbOq+be4JLfOELw08jJQintuk3kIXOPca06NMWQhGic3m7kRF95nM2TJs7
jqbkAMrKyizArcURMo0YpRZPIF7D1DlNRAebH2+0v0BxaUtombRDFW4U1SscuebXDndgjp7KjCnT
BJGDJks9/wLKKvFtISQWHuvN2MiPzt7UmFwMLPh8jtpgQ6JxS+ipTPxbr7Km3IXIJJgJHpxmdQmg
ghrl4IX0zCKaVUb7Rh3z85/9F0yPB8A92nquzQ==
recipientName: Test Business
paymentMethod: CC
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: PendingNetworkResponse
-----
```

## Sample IPN Success Notification to Rest Request

```
-----
```



```
transactionId: 14GKD9GE66FAA63E606B2JDPZKN53LZ7F22
statusMessage: The transaction was successful and the payment instrument was
  charged.
transactionDate: 1254815482
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference05
operation: SETTLE
transactionStatus: SUCCESS
transactionAmount: USD 1.00
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature: pwozZP
+lYONFq39g13ux44vFFMRAt4eJ9kOUWMV2uPCrvBqzi4LFYDQY5UE3VW8OUiW+qpbukqFz
YNvE+8mh7adhX/qee2U8ZUUNZi6LaM3sKtpPxus2ZJ3wDVPju002Obtu1G6Eo79iMi8viX7Dz1LL
8pFTdhspHZb0XDWkuOt2pK2aELa7TOZ/pXXUFLvGrn4M0d6INwbyM2fvnJpIDTcNdzedBO3Rw3vp
2f2GfpFAZJD6Imu57rsr9RsHVUqu2bIhJaAgTRFleVKzMHQJqft5jo6M9N4vKmpfccsuAvoF+rDn
+/6a9VEvTBrVcvAhJ5jrBp3FkXYkOPbHchqHfQ==
recipientName: Test Business
paymentMethod: CC
certificateUrl: https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: Success
-----
```

## Related Actions

- [Pay \(p. 74\)](#)
- [Refund \(p. 81\)](#)
- [Reserve \(p. 85\)](#)

## VerifySignature

### Description

`VerifySignature` enables you to verify the signature included with [outbound notifications](#). A correctly formatted call using `VerifySignature` returns a positive result when the signature is valid for the response that contained it.


This action is a component of signature version 2. Because of this, you may only use it with responses which have a `SignatureVersion` value of 2. To use signature version 2, you must select that option on your [Developer and Seller Preferences](#) page. .



#### Note

Because this action's purpose is simply to verify that a signature is valid and was generated by Amazon Payments, you do not need a developer account to invoke it from your code. Further, the request requires no signature.

## Request Parameters

Parameter	Description	Required
<i>UrlEndpoint</i>	<p>A required field that contains the appropriate originating endpoint (either the returnUrl or ipnUrl) that received the response. For example, if your web application resides at <code>http://my-app-website.biz/</code>, the returnUrl might be <code>http://my-app-website.biz/amazon/success.php</code>, and the IPNUrl might be <code>http://my-app-website.biz/amazon/ipnProcessor.php</code>.</p> <p>Type: String Default: None Constraint: Cannot be null or empty</p>	Yes
<i>HttpParameters</i>	<p>Concatenated string of all URL-Encoded parameters which were included in the response containing the signature you want to verify. This includes the <i>certificateUrl</i>, <i>signatureVersion</i>, <i>signatureMethod</i> and <i>signature</i> parameters.</p> <p>For example, a correctly formatted and URL-encoded string resembles the following:</p> <pre>First%20Name=Joe&amp;Last%20Name=Smith&amp;signatureVersion=2&amp;signatureMethod=HMACSHA256&amp;certificateUrl=https%253A%252F%252Ffps.amazonaws.com%252Fcert%252Fkey.pem&amp;signature=aoeuAOE123eAUdhf]</pre> <p> <b>Tip</b></p> <p>For validating the returnUrl, you can extract the query string from the returnUrl (excluding the '?' character). For validating the IPNUrl, concatenate the POST parameters.</p> <p>Type: String Default: None Constraint: Cannot be null or empty. In addition, because <i>VerifySignature</i> is a component of signature version 2, the value for <i>signatureVersion</i> must be 2.</p>	Yes

You must also use the *Action* parameter as described in [Common Request Parameters \(p. 53\)](#). Parameter names are case sensitive.

## Response Elements

Element	Description
<i>VerificationStatus</i>	<p>The result of the verification, either <i>Success</i> or <i>Failure</i>.</p> <p>Type: VerificationStatus</p>

Responses also include elements common to all responses. For more information, see [Common Response Elements](#) (p. 54).

## Errors

This action can return the following errors:

- [InternalServerError](#)
- [InvalidParams](#) (p. 58)

## Examples

### Sample REST Request

This section shows a sample request (SAMPLES TBD).

```
https://fps.sandbox.amazonaws.com?
Action=VerifySignature
&HttpParameters=signatureVersion%3D2%26signatureMethod%3DRSA-SHA1%26status%3DSubscriptionCancelled%26signature%3DsNaPePlaNg5pjeHYJ97BAPWoZVPxFpXGOMDmprYkPq8KN1cuZotBW2j%252BgoUqA5tue%252F2FD1nk5%252BZKMBtshSLiqtG1R6AH9qaNjZQwg4dm4t0OqP2eOjoH73wQwIaCCEr690o2lxjN%252Bvx7KO%252Bw4wmnyqFxl9%252Fj5wBjC2zpy3NrN8uM0R547rYjjOaTODYb0cesYfvXXPGvFBniDloPGpxx7G2ryIVZpFaeJ92XF2k6ho8M8rkdTp3MHPLiyZHjF16%252BcKen2XynOqHD5RkG%252FaIgG9wauS3E3esn9Zweo8m4vdiL67MyS4zQzyRg973bi45%252BKnv6AuuhhcTta41zSR8g%253D%253D%26subscriptionId%3D17d62772-c53e-4bdb-9667-65d7b7841cfc%26certificateUrl%3Dhttps%253A%252F%252Ffps.sandbox.amazonaws.com%252Fcerts%252F090909%252FFPKICert.pem%26statusReason%3DCancelledByRecipient
&Timestamp=2009-10-06T09%3A09%3A54.140Z
&UrlEndPoint=http%3A%2F%2Fmywebsite.com%3A8080%2Fipn.jsp
```

### Sample Query Request

```
GET\n
fps.sandbox.amazonaws.com\n
Action=VerifySignature
&HttpParameters=signatureVersion%3D2%26signatureMethod%3DRSA-SHA1%26status%3DSubscriptionCancelled%26signature%3DsNaPePlaNg5pjeHYJ97BAPWoZVPxFpXGOMDmprYkPq8KN1cuZotBW2j%252BgoUqA5tue%252F2FD1nk5%252BZKMBtshSLiqtG1R6AH9qaNjZQwg4dm4t0OqP2eOjoH73wQwIaCCEr690o2lxjN%252Bvx7KO%252Bw4wmnyqFxl9%252Fj5wBjC2zpy3NrN8uM0R547rYjjOaTODYb0cesYfvXXPGvFBniDloPGpxx7G2ryIVZpFaeJ92XF2k6ho8M8rkdTp3MHPLiyZHjF16%252BcKen2XynOqHD5RkG%252FaIgG9wauS3E3esn9Zweo8m4vdiL67MyS4zQzyRg973bi45%252BKnv6AuuhhcTta41zSR8g%253D%253D%26subscriptionId%3D17d62772-c53e-4bdb-9667-65d7b7841cfc%26certificateUrl%3Dhttps%253A%252F%252Ffps.sandbox.amazonaws.com%252Fcerts%252F090909%252FFPKICert.pem%26statusReason%3DCancelledByRecipient
&Timestamp=2009-10-06T09%3A09%3A54.140Z
&UrlEndPoint=http%3A%2F%2Fmywebsite.com%3A8080%2Fipn.jsp
```

### Sample Response to REST Request

This section shows a sample REST response.

```
<VerifySignatureResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <VerifySignatureResult>
    <VerificationStatus>Success</VerificationStatus>
```

```
</VerifySignatureResult>
<ResponseMetadata>
  <RequestId>197e2085-1ed7-47a2-93d8-d76b452acc74:0</RequestId>
</ResponseMetadata>
</VerifySignatureResponse>
```

## Data Types

This section describes the data types common to the Amazon FPS actions.

- [Enumerated Data Types \(p. 96\)](#)
- [Complex Data Types \(p. 101\)](#)

## Enumerated Data Types

### Topics

- [AccountBalance \(p. 96\)](#)
- [ChargeFeeTo \(p. 97\)](#)
- [CurrencyCode \(p. 97\)](#)
- [FPSOperation \(p. 97\)](#)
- [InstrumentId \(p. 98\)](#)
- [InstrumentStatus \(p. 98\)](#)
- [PaymentMethod \(p. 98\)](#)
- [RelationType \(p. 99\)](#)
- [SortOrder \(p. 99\)](#)
- [TokenStatus \(p. 99\)](#)
- [TokenType \(p. 99\)](#)
- [TransactionalRole \(p. 100\)](#)
- [TransactionStatus \(p. 100\)](#)

This section describes the enumerated data types Amazon FPS uses.

## AccountBalance

Name	Description	Type
<i>AvailableBalances</i>	The total amount of money that is transferred to your account from a bank account transfer or a refund.	<a href="#">AvailableBalances (p. 102)</a>
<i>PendingInBalance</i>	The total amount that is yet to be credited to your account.	<a href="#">Amount (p. 101)</a>
<i>PendingOutBalance</i>	The total amount that is yet to be debited from your account.	<a href="#">Amount (p. 101)</a>
<i>TotalBalance</i>	The total balance that is currently available in your account.	<a href="#">Amount (p. 101)</a>

## ChargeFeeTo

Name	Description	Type
<i>Caller</i>	Caller shall pay the fees.	String
<i>Recipient</i>	Recipient shall pay the fees.	String

## CurrencyCode

Name	Description	Type
<i>USD</i>	The transaction uses U.S. dollars.	String

## FPSOperation

These values are returned for non-IPN operations.

Name	Description	Type
<i>Pay</i>	All pay transactions.	String
<i>Refund</i>	All refund transactions.	String
<i>Settle</i>	All settle transactions.	String
<i>SettleDebt</i>	All debt settlement transactions.	String
<i>WriteOffDebt</i>	All debt write-off transactions.	String
<i>FundPrepaid</i>	All funding of prepaid transactions.	String
<i>Reserve</i>	All reserve transactions.	String

These values are returned only for IPN operations.

Name	Description	Type
<i>PAY</i>	All pay transactions.	String
<i>REFUND</i>	All refund transactions.	String
<i>SETTLE</i>	All settle transactions.	String
<i>SETTLE_DEBT</i>	All debt settlement transactions.	String
<i>WRITE_OFF_DEBT</i>	All debt write-off transactions.	String
<i>FUND_PREPAID</i>	All funding of prepaid transactions.	String
<i>RESERVE</i>	All reserve transactions.	String

Name	Description	Type
<i>MULTI_SETTLE</i>	All multi-settle transactions.	String
<i>REAUTH</i>	All transactions that required reauthorization.	String
<i>DEPOSIT_FUNDS</i>	All fund deposit transactions.	String
<i>WITHDRAW_FUNDS</i>	All fund withdrawal transactions.	String
<i>CANCEL_TRANSACTION</i>	All non-user cancelled transactions.	String
<i>CANCEL</i>	All user cancelled transactions.	String

## InstrumentId

Name	Description	Type
<i>InstrumentId</i>	An alphanumeric value that represents the payment instrument.	String Max size = 64 characters

## InstrumentStatus

Name	Description	Type
<i>Active</i>	All active instruments installed for your application.	String
<i>All</i>	All instruments installed for your application.	String
<i>Cancelled</i>	All canceled instruments.	String

## PaymentMethod

Name	Description	Type
<i>ABT</i>	Amazon Payments account balance transfer.	String
<i>ACH</i>	Bank account transaction.	String
<i>CC</i>	Credit card transaction.	String
<i>Debt</i>	Transactions using a credit instrument as payment method.	String
<i>Prepaid</i>	Transactions using a prepaid instrument as payment method.	String

## RelationType

Name	Description	Type
<i>MarketplaceFee</i>	Marketplace fee transactions.	String
<i>Parent</i>	Parent transactions.	String
<i>Refund</i>	Refund transactions.	String
<i>RefundReversal</i>	RefundReversal transactions.	String
<i>Reserve</i>	Reserve transactions.	String
<i>Settle</i>	Settle transactions.	String

## SortOrder

Name	Description	Type
<i>Ascending</i>	Return results in ascending order by date.	String
<i>Descending</i>	Return results in descending order by date (default).	String

## TokenStatus

Name	Description	String
<i>Active</i>	The token is in active state.	String
<i>Inactive</i>	The token was canceled by the user and is inactive.	String

## TokenType

Name	Description	Type
<i>MultiUse</i>	Token that can be used multiple times.	String
<i>Recurring</i>	Token which is specifically marked for recurring payments.	String
<i>SingleUse</i>	Token that can be used only once.	String

Name	Description	Type
<i>Unrestricted</i>	Token with unrestricted usage. Sender tokens with unlimited usage cannot be installed by external applications. Only recipient tokens can be installed with unrestricted usage.	String

## TransactionalRole

Name	Description	Type
<i>Caller</i>	Role is the caller.	String
<i>Recipient</i>	Role is the recipient.	String
<i>Sender</i>	Role is the sender.	String

## TransactionStatus

These values are returned for non-IPN operations.

Name	Description	Type
<i>Cancelled</i>	The transaction was canceled.	String
<i>Failure</i>	The transaction failed. The API operation failed and Amazon FPS did not receive or record a transaction. You can retry the transaction only if a retrievable error was returned.	String
<i>Pending</i>	The transaction is pending.	String
<i>Reserved</i>	The reserve request on the transaction succeeded. Amazon FPS reserves the purchase price against the sender's payment instrument.	String
<i>Success</i>	The transaction succeeded. You can fulfill the order for the customer.	String

## TransactionStatus (IPN)

These values are returned for IPN operations only.

Name	Description	Type
<i>CANCELLED</i>	The transaction was canceled.	String



Name	Description	Type
<i>FAILURE</i>	The transaction failed. The API operation failed and Amazon FPS did not receive or record a transaction. You can retry the transaction only if a retrievable error has been returned.	String
<i>PENDING</i>	The transaction is pending.	String
<i>RESERVED</i>	The reserve request on the transaction succeeded. Amazon FPS reserves the purchase price against the sender's payment instrument.	String
<i>SUCCESS</i>	The transaction succeeded. You can fulfill the order for the customer.	String

## Complex Data Types

### Topics

- [Amount](#) (p. 101)
- [AvailableBalances](#) (p. 102)
- [DebtBalance](#) (p. 102)
- [DescriptorPolicy](#) (p. 102)
- [MarketplaceRefundPolicy](#) (p. 102)
- [OutstandingDebtBalance](#) (p. 103)
- [OutstandingPrepaidLiability](#) (p. 103)
- [PrepaidBalance](#) (p. 103)
- [RelatedTransaction](#) (p. 103)
- [StatusHistory](#) (p. 104)
- [Token](#) (p. 104)
- [TokenUsageLimit](#) (p. 104)
- [Transaction](#) (p. 105)
- [TransactionDetail](#) (p. 106)
- [TransactionPart](#) (p. 108)

This section describes the complex data types Amazon FPS uses.

### Amount

Name	Description	Type
<i>CurrencyCode</i>	The currency code of the amount. Amazon FPS currently supports only USD.	<a href="#">CurrencyCode</a> (p. 97)
<i>Value</i>	The numeric value of the amount in dollars, for example, 25.00 is \$25, and 2500 is \$2500.	String

## AvailableBalances

Name	Description	Type
<i>DisburseBalance</i>	The total balance that has been disbursed.	Amount (p. 101)
<i>RefundBalance</i>	The total amount that has been refunded.	Amount (p. 101)

## DebtBalance

Name	Description	Type
<i>AvailableBalance</i>	Available debt balance accumulated between recipient and sender.	Amount (p. 101)
<i>PendingOutBalance</i>	Any balance that is pending because of an external instrument was used to settle the debt.	Amount (p. 101)

## DescriptorPolicy

For information about using the DescriptorPolicy type, see [Soft Descriptor Customization \(p. 45\)](#).

Name	Description	Type
<i>CSOwner</i>	The recipient or caller customer service number. If you specify <i>Caller</i> , the customer service number for the caller is passed to the payment processor, which is the entity that actually processes payments on the person's credit card or bank account. Otherwise, the default value of <i>CSOwner</i> is <i>Recipient</i> .	The entity whose CS Phone number should be used. Valid values are either <i>Recipient</i> or <i>Caller</i> . For more information, see <a href="#">Soft Descriptor Customization (p. 45)</a> .  Default: <i>Recipient</i>
<i>SoftDescriptorType</i>	The type of soft descriptor. Valid values are either <i>Static</i> or <i>Dynamic</i> . If you specify <i>Static</i> , or do not specify a type, the soft descriptor in your account level setting is sent to the payment processor. If you specify <i>Dynamic</i> , the first 15 characters of sender description is sent to the payment processor.	The type of soft descriptor. Valid values are either <i>Static</i> or <i>Dynamic</i> .  Default: <i>Static</i>

## MarketplaceRefundPolicy

Name	Description	Type
<i>MarketplaceTxnOnly</i>	Caller refunds his fee to the recipient.	String

Name	Description	Type
<i>MasterAndMarketplaceTxn</i>	Caller and Amazon FPS refund their fees to the sender, and the recipient refunds his amount	String
<i>MasterTxnOnly</i>	Caller does not refund his fee. Amazon FPS refunds its fee and the recipient refunds his amount plus the caller's fee to the sender.  Type: String	String

## OutstandingDebtBalance

Name	Description	Type
<i>OutstandingBalance</i>	Available debt balance accumulated between recipient and sender.	<a href="#">Amount (p. 101)</a>
<i>PendingOutBalance</i>	Any balance that is pending because an external instrument was used to settle the debt.	<a href="#">Amount (p. 101)</a>

## OutstandingPrepaidLiability

Name	Description	Type
<i>OutstandingBalance</i>	Outstanding prepaid liability owed by this account to all the senders who bought prepaid instruments.	<a href="#">Amount (p. 101)</a>
<i>PendingInBalance</i>	Any transient balance that is pending and yet to be settled.	<a href="#">Amount (p. 101)</a>

## PrepaidBalance

Name	Description	Type
<i>AvailableBalance</i>	Available prepaid balance funded by the sender to pay a particular recipient.	<a href="#">Amount (p. 101)</a>
<i>PendingInBalance</i>	Any balance that is pending because an external instrument is used to fund the instrument.	<a href="#">Amount (p. 101)</a>

## RelatedTransaction

Name	Description	Type
<i>RelationType</i>	Relation type of the related transaction.	<a href="#">RelationType (p. 99)</a>
<i>TransactionId</i>	The Transaction ID of the related transaction.	String  Max size = 35 characters

## StatusHistory

Name	Description	Type
<i>Amount</i>	The changed amount.	<a href="#">Amount (p. 101)</a>
<i>Date</i>	The date when the status changed.	dateTime
<i>StatusCode</i>	The current status of the transaction.	String
<i>TransactionStatus</i>	The current status of the transaction.	<a href="#">TransactionStatus (p. 100)</a>

## Token

Name	Description	Type
<i>CallerReference</i>	Account ID of the caller who initiated the original request.	String Max size = 128 bytes
<i>DateInstalled</i>	The date and time when the payment token was created on the caller's account.	dateTime
<i>FriendlyName</i>	A name that references the token.	String Max size = 128 characters
<i>OldTokenId</i>	The token ID linked to this token. The token that was created in place of this token.	String
<i>PaymentReason</i>	Payment reason passed during token installation.	String
<i>TokenId</i>	The token ID representing the payment instruction.	String Max size = 64 characters
<i>TokenStatus</i>	Specifies whether or not the token is active.	<a href="#">TokenStatus (p. 99)</a>
<i>TokenType</i>	The type of the token (e.g., single-use, multi-use, etc.).	<a href="#">TokenType (p. 99)</a>

## TokenUsageLimit

Name	Description	Type
<i>Amount</i>	Amount paid in the latest time window with this token.	<a href="#">Amount (p. 101)</a>
<i>Count</i>	Number of times this token was used in the latest time window.	Integer
<i>LastResetAmount</i>	Amount paid in the previous time window with this token.	<a href="#">Amount (p. 101)</a>

Name	Description	Type
<i>LastResetCount</i>	Number of times this token was used in the previous time window.	Integer
<i>LastResetTimeStamp</i>	The exact time when the latest time window started for this limit.	dateTime

## Transaction

Name	Description	Type
<i>Balance</i>	Balance in prepaid account.	<a href="#">Amount (p. 101)</a>
<i>CallerName</i>	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
<i>CallerTransactionDate</i>	Date the caller provided for the transaction.	dateTime
<i>DateCompleted</i>	Date the transaction was completed.	dateTime
<i>DateReceived</i>	Date the transaction was received by Amazon FPS.	dateTime
<i>FPSFees</i>	Amount of fees collected by Amazon FPS for performing the transaction.	<a href="#">Amount (p. 101)</a>
<i>FPSOperation</i>	The operation type.	<a href="#">FPSOperation (p. 97)</a>
<i>OriginalTransactionId</i>	In the case of a refund, the TransactionID that is being reversed.	String Max size = 35 characters
<i>PaymentMethod</i>	Payment method used in the transaction.	<a href="#">Payment Method (p. 98)</a>
<i>RecipientName</i>	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
<i>RecipientTokenID</i>	The recipient token used in the transaction.	String
<i>SenderName</i>	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
<i>SenderTokenID</i>	The sender token used in the transaction.	String

Name	Description	Type
<i>StatusCode</i>	A code that represents the current status of the transaction. Expands on the information in the <i>TransactionStatus</i> field. For example, if <i>TransactionStatus</i> is PENDING, this field might be <i>PendingVerification</i> , or <i>PendingNetworkResponse</i> .	String
<i>StatusMessage</i>	A short description of the current status of the transaction.	String
<i>TransactionAmount</i>	Total amount of the transaction.	<a href="#">Amount (p. 101)</a>
<i>TransactionId</i>	Unique Amazon FPS-generated ID for the transaction.	String Max size = 35 characters
<i>TransactionPart</i>	List of individual parts of the transaction, with each one dealing with your account's role in the transaction.	<a href="#">TransactionPart (p. 108)</a>
<i>TransactionStatus</i>	Provides a short code on the status of the transaction, for example "PENDING".	<a href="#">Transaction Status (p. 100)</a>

## TransactionDetail

Name	Description	Type
<i>CallerName</i>	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
<i>CallerDescription</i>	Caller description the caller provided for the transaction.	String
<i>CallerReference</i>	Caller reference the caller provided for the transaction.	String
<i>CreditInstrumentID</i>	In the case of a postpaid transaction, this is the credit instrument ID.	String
<i>DateReceived</i>	Date Amazon FPS received the transaction.	dateTime
<i>DateCompleted</i>	Date the transaction was completed.	dateTime
<i>FPSFees</i>	Amount of fees collected by Amazon FPS for performing the transaction.	<a href="#">Amount (p. 101)</a>
<i>FPSFeesPaidBy</i>	The party paying the FPS fees for this transaction.	<a href="#">TransactionalRole (p. 100)</a>
<i>FPSOperation</i>	The operation type.	<a href="#">FPSOperation (p. 97)</a>

**Amazon FPS Advanced Quick Start Developer Guide**  
**Complex Data Types**

<b>Name</b>	<b>Description</b>	<b>Type</b>
<i>MarketPlaceFees</i>	In the case of a marketplace transaction, this is the amount of any marketplace fee the caller has charged.	<a href="#">Amount (p. 101)</a>
<i>PaymentMethod</i>	The payment method used.	<a href="#">PaymentMethod (p. 98)</a>
<i>PrepaidInstrumentID</i>	In the case of a prepaid transaction, this is the prepaid instrument ID.	String
<i>RecipientEmail</i>	The e-mail ID of the recipient of this transaction.	String
<i>RecipientName</i>	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
<i>RecipientTokenId</i>	Recipient token ID used in the transaction.	String
<i>RelatedTransaction</i>	All transactions related to this transaction.	<a href="#">RelatedTransaction (p. 103)</a>
<i>SenderDescription</i>	Sender description the caller provided for the transaction.	String
<i>SenderEmail</i>	The email ID of the sender of this transaction. This is returned only if the caller is also the recipient of this transaction.	String
<i>SenderName</i>	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
<i>SenderTokenId</i>	Sender token ID used in the transaction.	String
<i>StatusCode</i>	A code that represents the current status of the transaction.	String
<i>StatusHistory</i>	A list of all the previous status entries for this transaction.	<a href="#">StatusHistory (p. 104)</a>
<i>StatusMessage</i>	A short description of the current status of the transaction.	String
<i>TransactionAmount</i>	Total amount of the transaction.	<a href="#">Amount (p. 101)</a>
<i>TransactionId</i>	Unique Amazon FPS-generated ID for the transaction.	String Max size = 35 characters
<i>TransactionStatus</i>	The transaction status.	<a href="#">TransactionStatus (p. 100)</a>

## TransactionPart

Name	Description	Type
<i>Description</i>	Description provided by the entity.	String
<i>FeesPaid</i>	Fees the caller or recipient paid.	<a href="#">Amount (p. 101)</a>
<i>InstrumentId</i>	Payment instrument involved in this transaction part.	String
<i>Name</i>	Name used for the role specified in <i>Role</i> .	String
<i>Reference</i>	Reference data provided by this party.	String
<i>Role</i>	Role played by this party.	<a href="#">TransactionalRole (p. 100)</a>



# Co-Branded Service API Reference

---

## Topics

- [Common Parameters](#) (p. 109)
- [Recipient Token API](#) (p. 112)
- [Recurring-Use Token API](#) (p. 114)
- [Multi-Use Token API](#) (p. 117)
- [Edit Token API](#) (p. 122)

You use different Co-Branded service APIs to create different payment tokens. For example, the Recurring-Use Token API creates a recurring-use payment token, whereas the Multi-Use Token API creates a multi-use payment token. The Advanced Quick Start also includes the Recipient Token API, which registers merchants with you (rather than creating a payment token) so they can receive payments. For more information about the different tokens, see [Multi-Use Payment Tokens](#) (p. 9), [Recurring Payment Tokens](#) (p. 11), and [Recipient Tokens](#) (p. 13).

## Common Parameters


The following parameters are common to all Co-Branded service API requests.

### Request Parameters

Name	Description	Required
<i>callerKey</i>	AWS Access Key ID of the developer. You can obtain this value from the AWS Access Identifiers page on the AWS web site ( <a href="http://aws.amazon.com">http://aws.amazon.com</a> ). Type: String Default: None	Yes

**Amazon FPS Advanced Quick Start Developer Guide**  
**Request Parameters**

<b>Name</b>	<b>Description</b>	<b>Required</b>
<i>cobrandingStyle</i>	<p>Specifies the co-branding type on Amazon FPS payment authorization pages. Amazon FPS is phasing out support for the banner type, so we suggest you change your co-branding to the logo type.</p> <p>For more information, see <a href="#">Co-Branding Styles (p. 111)</a>.</p> <p>Type: String            Default: <code>logo</code>            Valid Values: <code>banner</code>   <code>logo</code></p>	No
<i>cobrandingUrl</i>	<p>Allows you to specify a co-branding URL dynamically. It specifies the URL of your company's logo.</p> <p>Type: String            Default: None</p> <p>Constraint: This URL should point to a co-branding image that is not larger than 215 (w) x 40 (h) pixels in a secure HTTP server.</p>	No
<i>pipelineName</i>	<p>The kind of token you are creating.</p> <p>Type: String            Default: None</p> <p>Valid Values: <code>SingleUse</code>   <code>MultiUse</code>   <code>Recurring</code>   <code>Recipient</code>   <code>SetupPrepaid</code>   <code>SetupPostpaid</code>   <code>EditToken</code></p>	Yes
<i>returnURL</i>	<p>Specifies the URL on your web site that the person (typically the buyer) is redirected to after completing the CBUI web pages. In addition to the URL, the redirect URI includes the following:</p> <ul style="list-style-type: none"> <li>Parameters appended to the <i>returnURL</i> in the URI</li> <li>Status of the request</li> <li>The installed token</li> <li>The <i>Signature</i></li> </ul> <p>Type: URL            Default: None</p>	Yes
<i>signature</i>	<p>A value calculated using the request parameters and a SHA-1 HMAC encryption algorithm to make sure the request parameters and values were not altered during the request's or response's travel across the Internet. For more information, see <a href="#">Working with Signatures (p. 41)</a>.</p> <p>Type: String            Default: None</p>	Yes

Name	Description	Required
<i>signatureVersion</i>	1 or 2   <b>Caution</b> If you are currently using signature version 1: Version 1 is deprecated, and you should move to signature version 2 as soon as possible. For information about the deprecation schedule and the differences between signature version 2 and version 1, go to <a href="#">Making Secure Requests to Amazon Web Services</a> .	Yes
<i>signatureMethod</i>	<i>HmacSHA256</i> (preferred) or <i>HmacSHA1</i>	Yes
<i>version</i>	The version of the API to use. Always set to 2009-01-09. Type: String	Yes
<i>websiteDescription</i>	Human readable text to describe your web site. It is used on the payment authorization pages for messaging only. For instance, a message such as "Click here to return to <websiteDescription> website" can appear on the page. Type: String Default: None	No

## Co-Branding Styles

Co-branding refers to using your brand along with Amazon's on the CBUI pages. The CBUI offers the following co-branding styles.

- **Banner**—Your logo appears in the upper left corner of the CBUI page and the Amazon Payments logo appears right below your logo on the right hand side. Amazon is phasing out support for the banner type in favor of the logo type.
- **Logo**—Your logo appears on the upper left corner of the CBUI page, followed by a checkout cart breadcrumb in the middle, followed by the Amazon Payments logo, as shown.  
This is the default behavior.

The following figure shows an example of banner co-branding.



The following figure shows an example of logo co-branding.



## Response Parameters

The following table lists the parameters common to all Co-Branded service API responses.

Name	Description
<i>Signature</i>	<p>Amazon FPS calculates the <i>Signature</i> using all the parameters in the <i>returnURL</i>. We recommend that you calculate the return URL's <i>Signature</i> using the same method that you used to calculate the <i>Signature</i> for your signed URL. This is to ensure that you are receiving the response from Amazon FPS.</p> <p>For more information, see <a href="#">Working with Signatures (p. 41)</a>.</p> <p>Type: String</p>

## Recipient Token API

Use this Co-Branded service API to register a recipient on a caller's web site.

### Request Parameters

Parameter	Description	Required
<i>callerReference</i>	<p>A value you provide that uniquely identifies the request. For more information, see <a href="#">Important Values to Store in Your Database (p. 30)</a>.</p> <p>Type: String</p> <p>Constraint: Max size = 128 bytes</p>	Yes
<i>maxFixedFee</i>	<p>The maximum fixed fee that the caller charges in a marketplace transaction. The actual fixed marketplace fee is passed as a parameter to <code>Pay</code> or <code>Reserve</code> actions. This field can be ignored if a fixed fee is not being charged.</p> <p>Type: Long</p>	Yes
<i>maxVariableFee</i>	<p>The maximum variable fee that the caller charges in a marketplace transaction. The variable fee is a percentage of the transaction amount. The actual variable marketplace fee is passed as a parameter to <code>Pay</code> or <code>Reserve</code> actions. This field can be ignored if a variable fee is not being charged.</p> <p>Type: Long</p>	Yes
<i>paymentMethod</i>	<p>Specifies payment methods the recipient supports. Use <code>CC</code> for credit cards, <code>ACH</code> for bank account withdrawal, and <code>ABT</code> for Amazon Payments balance transfer.</p> <p>Type: Comma-separated list</p> <p>Default: <code>ABT</code></p> <p>Valid Values: <code>CC   ACH   ABT</code></p>	No
<i>recipientPaysFee</i>	<p>Set this value to <code>True</code> if the recipient agrees to pay the fees, otherwise set this value to <code>False</code>.</p> <p>Type: String</p> <p>Valid Values: <code>True   False</code></p>	Yes
<i>validityExpiry</i>	<p>When the token expires.</p> <p>Type: Date</p> <p>Default: No expiry</p> <p>Constraint: Date cannot be earlier than the current date</p>	No

Parameter	Description	Required
<i>validityStart</i>	When the token becomes valid. Provides seconds from the EPOCH time. Type: Date Default: Current date Constraint: Date must be within one year from the current date (date of creation) and cannot be earlier than the current date	No



#### Note

Co-Branded service request parameters are not case sensitive.

The request also uses the parameters common to all Co-Branded service API requests. For more information, see [Common Parameters \(p. 109\)](#).

## Response Parameters

Parameter	Description
<i>errorMessage</i>	This is text in a human readable form that specifies the reason for a request failure. Type: String
<i>status</i>	Specifies the status of the Co-Branded service request. Type: String Valid values: See the following table
<i>tokenID</i>	This string identifies the merchant who gets paid in the transaction. You should store this value. Type: String

Responses also include parameters common to all responses. For more information, see [Response Parameters \(p. 111\)](#).

## Status Code

The following table shows the values of the *status* response parameter.

Status Code	Description
SR	Success. Specifies that the merchant's token was created.
A	Specifies that the pipeline has been aborted by the user.
CE	Specifies a caller exception.

Status Code	Description
NP	<p>There are four cases where the NP status is returned:</p> <ul style="list-style-type: none"> <li>• The payment instruction installation was not allowed on the sender's account, because the sender's email account is not verified</li> <li>• The sender and the recipient are the same</li> <li>• The recipient account is a personal account, and therefore cannot accept credit card payments</li> <li>• A user error occurred because the pipeline was cancelled and then restarted</li> </ul>
NM	You are not registered as a third-party caller to make this transaction. Contact Amazon Payments for more information.

## Recurring-Use Token API

The recurring-use token API creates a token that makes payments on a recurring basis at specific intervals. The sender's account is charged accordingly without requiring recurring authorizations. When the *recurringPeriod* is one month, the payment occurs on the same day each month. If the original payment occurred on the last day of a month, the next payment occurs on the closest day to that without skipping a month. For example, if the pay date is October 31, the next pay date will be November 30. After the recurrence passes through February, the pay date is typically on the 28th of each month.

### Request Parameters

Parameter	Description	Required
<i>addressName</i> <i>addressLine1</i> <i>addressLine2</i> <i>city</i> <i>state</i> <i>zip</i> <i>phoneNumber</i>	<p>The sender's shipping address. You might choose to collect the address on your web site and pass it to the CBUI. If you choose to collect the shipping address yourself, you can use these parameters to specify it. This address will be displayed to the sender on the payment authorization confirmation page. See also the description of <i>collectShippingAddress</i>.</p> <p>Type: String Default: None</p>	No
<i>callerReference</i>	<p>A value you provide that uniquely identifies the request. For more information, see <a href="#">Important Values to Store in Your Database (p. 30)</a>.</p> <p>Type: String Default: None Constraint: Max size = 128 bytes</p>	Yes
<i>collectShippingAddress</i>	<p>If you set this value to <code>True</code>, all the shipping address parameters (<i>addressName</i>, <i>addressLine1</i>, etc.) are ignored, and the shipping/mailling address that the sender confirms on the CBUI pages is returned as part of the return URL.</p> <p>Type: Boolean Default: <code>False</code> Valid Values: <code>True</code>   <code>False</code></p>	No

**Amazon FPS Advanced Quick Start Developer Guide**  
**Request Parameters**

<b>Parameter</b>	<b>Description</b>	<b>Required</b>
<i>currencyCode</i>	Specifies the currency of all amounts that this pipeline accepts. Type: String Default: USD Valid Values: USD	No
<i>isRecipientCobranding</i>	This parameter is for marketplace applications where the caller is different from the recipient. If this value is set to <code>True</code> , then the co-branding URL is picked up from the recipient's settings. The parameter <i>recipientToken</i> is mandatory if this value is set to <code>True</code> . If this value is <code>False</code> , the co-branding URL is picked up from the caller's setting. Type: Boolean Valid Values: <code>True</code>   <code>False</code> Default: <code>False</code>	No
<i>paymentMethod</i>	Specifies payment methods the recipient supports. Use <code>CC</code> for credit cards, <code>ACH</code> for bank account withdrawal, and <code>ABT</code> for Amazon Payments balance transfer. Type: Comma-separated list Default: <code>ABT</code> Valid Values: <code>CC</code>   <code>ACH</code>   <code>ABT</code>	No
<i>paymentReason</i>	Specifies the reason for this payment transaction. You can provide a limited set of HTML tags to format your text, including <code>&lt;b&gt;</code> , <code>&lt;i&gt;</code> , <code>&lt;u&gt;</code> , <code>&lt;ul&gt;</code> , <code>&lt;li&gt;</code> , <code>&lt;br&gt;</code> , <code>&lt;em&gt;</code> , <code>&lt;strong&gt;</code> , and <code>&lt;strike&gt;</code> . Other tags are ignored. Type: String	No
<i>recipientToken</i>	Specifies the intended recipient's <i>TokenId</i> . Type: String Default: The caller is considered to be the recipient	No
<i>recurringPeriod</i>	The recurring period to associate with this token. This parameter accepts an integer followed by one of the following strings: <ul style="list-style-type: none"> <li>Hour[s] (e.g. 12 Hours)</li> <li>Day[s] (e.g., 4 Days)</li> <li>Month[s] (e.g., 6 Months)</li> </ul> Type: String	Yes
<i>transactionAmount</i>	Specifies the recurring amount payable in this transaction. Type: String	Yes
<i>validityExpiry</i>	When the token expires. The expiry date has no restrictions, other than the date cannot be earlier than the current date. Type: Date Default: No expiration	No

Parameter	Description	Required
<i>validityStart</i>	When the token becomes valid. By default this date is current date. Provides seconds unit using the EPOCH time. Type: Date Default: The current date Constraint: The validity start date should be within one year from the current date. The date cannot be beyond one year or earlier than the current date.	No

The request also uses the parameters common to all Co-Branded service API requests. For more information, see [Common Parameters \(p. 109\)](#).

## Response Parameters

Parameter	Description
<i>addressName</i> <i>addressLine1</i> <i>addressLine2</i> <i>city</i> <i>state</i> <i>zip</i> <i>phoneNumber</i>	The sender's shipping address. These parameters are returned only if <i>collectShippingAddress</i> was set to <code>True</code> in the request. Type: String
<i>errorMessage</i>	This is text in a human readable form that specifies the reason for a request failure. Type: String
<i>expiry</i>	The expiry (if any) of the payment method. Type: String
<i>status</i>	The status of the Co-Branded service request. Type: String Valid Values: See <a href="#">Status Codes (p. 117)</a> .
<i>tokenID</i>	Specifies the token ID string associated with the token just created (installed). Type: String
<i>warningCode</i>	There might be cases when the sender token is installed successfully but there is an associated warning. This parameter denotes that warning. Type: String Valid Values: <i>invalidShippingAddress</i> (returned when you pass an incorrect shipping address in the request parameters <i>addressLine1</i> , <i>addressLine2</i> , <i>city</i> , etc.)
<i>warningMessage</i>	Specifies a human readable text that explains the warning corresponding to the <i>warningCode</i> . Type: String



Responses also include parameters common to all responses. For more information, see [Response Parameters](#) (p. 111).

## Status Codes

Status Code	Description
SA	Success status for the ABT payment method.
SB	Success status for the ACH (bank account) payment method.
SC	Success status for the credit card payment method.
SE	System error.
A	Buyer abandoned the pipeline.
CE	Specifies a caller exception.
PE	Payment Method Mismatch Error: Specifies that the buyer does not have payment method that you have requested.
NP	There are four cases where the NP status is returned: <ul style="list-style-type: none"><li>• The payment instruction installation was not allowed on the sender's account, because the sender's email account is not verified</li><li>• The sender and the recipient are the same</li><li>• The recipient account is a personal account, and therefore cannot accept credit card payments</li><li>• A user error occurred because the pipeline was cancelled and then restarted</li></ul>
NM	You are not registered as a third-party caller to make this transaction. Contact Amazon Payments for more information.

## Multi-Use Token API

The multi-use token API creates a token that makes multiple payments without the sender having to repeatedly authorize payments. The payments can occur at any time; they don't have to be on a regularly recurring basis. This token works like a voucher and the parameters you include in the request govern the token's use, such as how much the token can pay per transaction, how much it can pay during its lifetime, when it expires, what kind of personal payment instruments it can accept, and the minimum payment it can make.

You can specify a maximum amount that can be charged over a period of time. Set *usageLimitType1* to `Amount`, and then use *usageLimitPeriod1* and *usageLimitValue1* to set the time period and maximum amount.

You can also specify a maximum number of charges that can occur over a period of time. Set *usageLimitType2* to `Count`, and then use *usageLimitPeriod2* and *usageLimitValue2* to set the time period and maximum number of charges.

## Request Parameters

Parameter	Description	Required
<i>addressName</i> <i>addressLine1</i> <i>addressLine2</i> <i>city</i> <i>state</i> <i>zip</i> <i>phoneNumber</i>	The sender's shipping address. You might choose to collect the address on your web site and pass it to the CBU. If you choose to collect the shipping address yourself, you can use these parameters to specify it. This address will be displayed to the sender on the payment authorization confirmation page. See also the description of <i>collectShippingAddress</i> . Type: String Default: None	No
<i>amountType</i>	Specifies whether the amount specified by <i>transactionAmount</i> is the exact amount allowed by the token, the minimum amount, or the maximum amount. Type: String Default: <code>Exact</code> Valid Values: <code>Exact</code>   <code>Maximum</code>   <code>Minimum</code>	No
<i>callerReference</i>	A value you provide that uniquely identifies the request. For more information, see <a href="#">Important Values to Store in Your Database (p. 30)</a> . Type: String Default: None Constraint: Max size = 128 bytes	Yes
<i>collectShippingAddress</i>	If you set this value to <code>True</code> , all the shipping address parameters ( <i>addressName</i> , <i>addressLine1</i> , etc.) are ignored, and the shipping/mailling address that the sender confirms on the CBU pages is returned as part of the return URL. Type: Boolean Default: <code>False</code> Valid Values: <code>True</code>   <code>False</code>	No
<i>currencyCode</i>	Specifies the currency of all amounts that this pipeline accepts. Type: String Default: <code>USD</code> Valid Values: <code>USD</code>	No
<i>globalAmountLimit</i>	Specifies the maximum amount of money this token can be used for across all payments. Type: String	Yes

**Amazon FPS Advanced Quick Start Developer Guide**  
**Request Parameters**

Parameter	Description	Required
<i>isRecipientCobranding</i>	This parameter is for marketplace applications where the caller is different from the recipient. If this value is set to <code>True</code> , then the co-branding URL is picked up from the recipient's settings. The parameter <i>recipientToken</i> is mandatory if this value is set to <code>True</code> . If this value is <code>False</code> , the co-branding URL is picked up from the caller's setting. Type: Boolean Valid Values: <code>True</code>   <code>False</code> Default: <code>False</code>	No
<i>paymentMethod</i>	Specifies payment methods the recipient supports. Use <code>CC</code> for credit cards, <code>ACH</code> for bank account withdrawal, and <code>ABT</code> for Amazon Payments balance transfer. Type: Comma-separated list Default: <code>ABT</code> Valid Values: <code>CC</code>   <code>ACH</code>   <code>ABT</code>	No
<i>paymentReason</i>	Specifies the reason for this payment transaction. You can provide a limited set of HTML tags to format your text, including <code>&lt;b&gt;</code> , <code>&lt;i&gt;</code> , <code>&lt;u&gt;</code> , <code>&lt;ul&gt;</code> , <code>&lt;li&gt;</code> , <code>&lt;br&gt;</code> , <code>&lt;em&gt;</code> , <code>&lt;strong&gt;</code> , and <code>&lt;strike&gt;</code> . Other tags are ignored. Type: String	No
<i>recipientTokenList</i>	Specifies a comma-separated list of recipient token IDs (for the merchants who can receive payment with this token). If you create a merchant token list, you must include the recipient token ID for the caller in addition to any other tokens you add to the list. Type: String Default: Caller's recipient token ID Conditional: Required if <i>isRecipientCobranding</i> is <code>True</code>	Conditional
<i>transactionAmount</i>	Specifies the amount payable in this transaction. Type: Double Condition: Required if you have specified <i>amountType</i> .	Conditional
<i>usageLimitType1</i>	See the description at the beginning of this topic for how to use this field. Type: String Valid Values: <code>Amount</code>   <code>Count</code>	No

**Amazon FPS Advanced Quick Start Developer Guide**  
**Request Parameters**

Parameter	Description	Required
<i>usageLimitPeriod1</i>	<p>If you specify <i>usageLimitType1</i> but not <i>usageLimitPeriod1</i>, the usage period is forever. This parameter accepts an integer followed by one of the following strings:</p> <ul style="list-style-type: none"> <li>Hour[s] (e.g., 12 Hours)</li> <li>Day[s] (e.g., 4 Days)</li> <li>Month[s] (e.g., 6 Months)</li> </ul> <p>Type: String            Default: Forever</p>	No
<i>usageLimitValue1</i>	<p>Corresponds to the value for the parameter specified by <i>usageLimitType1</i>.            Type: String if <i>usageLimitType1</i>=Amount; integer if <i>usageLimitType1</i>=Count            Condition: Required if you specify <i>usageLimitType1</i>.</p>	Conditional
<i>usageLimitType2</i>	<p>See the description at the beginning of this topic for how to use this field.            Type: String            Valid Values: Amount   Count</p>	No
<i>usageLimitPeriod2</i>	<p>If you specify <i>usageLimitType2</i> but not <i>usageLimitPeriod2</i>, the usage period is forever. This parameter accepts an integer followed by one of the following strings:</p> <ul style="list-style-type: none"> <li>Hour[s] (e.g., 12 Hours)</li> <li>Day[s] (e.g., 4 Days)</li> <li>Month[s] (e.g., 6 Months)</li> </ul> <p>Type: String            Default: Forever</p>	No
<i>usageLimitValue2</i>	<p>Corresponds to the value for the parameter specified by <i>usageLimitType2</i>.            Type: String if <i>usageLimitType2</i>=Amount; integer if <i>usageLimitType2</i>=Count            Condition: Required if you specify <i>usageLimitType2</i>.</p>	Conditional
<i>validityExpiry</i>	<p>When the token expires. The expiry date has no restrictions, other than the date cannot be earlier than the current date.            Type: Date            Default: No expiration</p>	No

Parameter	Description	Required
<i>validityStart</i>	When the token becomes valid. By default this date is current date. Provides seconds unit using the EPOCH time. Type: Date Default: The current date Constraint: The validity start date should be within one year from the current date. The date cannot be beyond one year or earlier than the current date.	No

The request also uses the parameters common to all Co-Branded service API requests. For more information, see [Common Parameters \(p. 109\)](#).

## Response Parameters

Parameter	Description
<i>addressName</i> <i>addressLine1</i> <i>addressLine2</i> <i>city</i> <i>state</i> <i>zip</i> <i>phoneNumber</i>	The sender's shipping address. These parameters are returned only if <i>collectShippingAddress</i> was set to <code>True</code> in the request. Type: String
<i>errorMessage</i>	This is text in a human readable form that specifies the reason for a request failure. Type: String
<i>expiry</i>	Specifies the expiry (if any) of the payment method. Type: String
<i>status</i>	The status of the Co-Branded service request. Type: String Valid Values: See <a href="#">Status Codes (p. 122)</a> .
<i>tokenID</i>	Specifies the token ID string associated with the token just created (installed). Type: String
<i>warningCode</i>	There might be cases when the sender token is installed successfully but there is an associated warning. This parameter denotes that warning. Type: String Valid Values: <i>invalidShippingAddress</i> (returned when you pass an incorrect shipping address in the request parameters <i>addressLine1</i> , <i>addressLine2</i> , <i>city</i> , etc.)
<i>warningMessage</i>	Specifies a human readable text that explains the warning corresponding to the <i>warningCode</i> . Type: String

Responses also include parameters common to all responses. For more information, see [Response Parameters \(p. 111\)](#).

## Status Codes

Status Code	Description
SA	Success status for the ABT payment method.
SB	Success status for the ACH (bank account) payment method.
SC	Success status for the credit card payment method.
SE	System error.
A	Buyer abandoned the pipeline.
CE	Specifies a caller exception.
PE	Payment Method Mismatch Error: Specifies that the buyer does not have the payment method you requested.
NP	There are four cases where the NP status is returned: <ul style="list-style-type: none"> <li>The payment instruction installation was not allowed on the sender's account, because the sender's email account is not verified</li> <li>The sender and the recipient are the same</li> <li>The recipient account is a personal account, and therefore cannot accept credit card payments</li> <li>A user error occurred because the pipeline was cancelled and then restarted</li> </ul>
NM	You are not registered as a third-party caller to make this transaction. Contact Amazon Payments for more information.

## Edit Token API

The Edit Token API enables you to view an existing token's details and to change the payment instrument for the token.

## Request Parameters

Parameter	Description	Required
<i>callerReference</i>	A value you provide that uniquely identifies the request. For more information, see <a href="#">Important Values to Store in Your Database (p. 30)</a> . Type: String Default: None Constraint: Max size = 128 bytes	Yes
<i>paymentMethod</i>	A comma-separated list that enables you to pass payment methods supported by the merchant. Type: String Default: ABT Valid Values: CC   ACH   ABT	No

Parameter	Description	Required
<i>tokenID</i>	Specifies the token you want to edit. You can only edit multi-use or recurring tokens. Type: String Constraint: Max size = 65 characters	Yes

The request also uses the parameters common to all Co-Branded service API requests. For more information, see [Common Parameters \(p. 109\)](#).

## Response Parameters

Parameter	Description
<i>errorMessage</i>	This is text in a human readable form that specifies the reason for a request failure. Type: String
<i>expiry</i>	This specifies the expiry (if any) of the payment method associated with the token. Type: String
<i>status</i>	The status of the Co-Branded service request. Type: String Valid Values: See <a href="#">Status Codes (p. 123)</a> .
<i>tokenID</i>	This specifies the token ID associated with the new token. Type: String

Responses also include parameters common to all responses. For more information, see [Response Parameters \(p. 111\)](#).

## Status Codes

Status Code	Description
SU	The token has not changed.
SA	The status of the token that uses ABT as the payment method has changed.
SB	The token that uses ACH as the payment method has changed.
SC	The token that uses CC as the payment method has changed.
SE	System error.
A	Buyer abandoned the pipeline.
CE	Specifies a caller exception.
IT	Specifies an invalid token. The token passed is invalid, expired, or cannot be modified.
NA	Specifies that the user account does not exist.
UT	Specifies an unauthorized access to the token. The token does not belong to the user.

**Amazon FPS Advanced Quick Start Developer Guide**  
**Status Codes**

---

<b>Status Code</b>	<b>Description</b>
PE	Specifies a payment method mismatch error. The user does not have payment method that you requested.



# Code Samples

## Topics

- [Understanding the Amazon FPS Samples \(p. 126\)](#)
- [Understanding the Amazon CBUI Samples \(p. 129\)](#)
- [Understanding the IPNAndReturnURLValidation Sample \(p. 133\)](#)
- [Getting the Samples \(p. 135\)](#)

This appendix provides an overview of the Amazon Flexible Payments Service development libraries provided by Amazon. The sample code shows you how to implement most of the basic Amazon FPS functions. Packaged in four programming languages (C#, Java, Perl, and PHP), the development libraries are available from the [Amazon Web Services developer community](#), under the Amazon Flexible Payments Service category. Refer to the following table for specific sample packages.

Language	Location
C#	<a href="http://signature2-fps.s3.amazonaws.com/amazon-fps-2008-09-17-cs-library.zip">http://signature2-fps.s3.amazonaws.com/amazon-fps-2008-09-17-cs-library.zip</a>
Java	<a href="http://signature2-fps.s3.amazonaws.com/amazon-fps-2008-09-17-java-library.zip">http://signature2-fps.s3.amazonaws.com/amazon-fps-2008-09-17-java-library.zip</a>
Perl	<a href="http://signature2-fps.s3.amazonaws.com/amazon-fps-2008-09-17-perl-library.zip">http://signature2-fps.s3.amazonaws.com/amazon-fps-2008-09-17-perl-library.zip</a>
PHP	<a href="http://signature2-fps.s3.amazonaws.com/amazon-fps-2008-09-17-php-library.zip">http://signature2-fps.s3.amazonaws.com/amazon-fps-2008-09-17-php-library.zip</a>

Each package is updated for signature version 2, and contains both a development library and sample implementations of the Amazon FPS APIs. The development libraries enable you to

- Use the Co-Branded User Interface to create CBUI pipeline URLs
- Invoke any of the Amazon FPS APIs documented in this quickstart
- Generating request signatures

- Validate the content of return URL responses and IPN notifications

The following sections describe the code libraries in more detail, with an eye towards enabling you to build your applications quickly. If you want help building your first sample application using the development libraries, see "Making a Pay Request" in the [Amazon Flexible Payments Service Getting Started Guide](#)

## Understanding the Amazon FPS Samples

Amazon provides dozens of samples in four programming languages (C#, Java, Perl, and PHP) which show you how to perform numerous operation with Amazon FPS actions.

When you download a sample file, such as `amazon-fps-2008-09-17-java-library`, the `[package root]/src/com/amazonaws/fps/samples` folder contains sample classes showing how to invoke most Amazon FPS actions from your code (`[package root]` is the location you extracted your sample package).

Each sample describes its requirements in its `Readme.html` file, located at the package root. Typically, the entire library structure must be available to the compiler. For example, the `amazon-fps-2008-09-17-php-library.zip` file contains the `src/Amazon/FPS/Model` and `src/Amazon/FPS/Mock` folders, which the files in `src/Amazon/FPS/Samples` require.

In addition to these primary components, a sample may include other required resources. For example, the Java samples all include numerous jar files in the `[package-root]/third-party` folder, which must also be in your classpath in order to compile the sample.

For each sample, you must set your security credentials and Amazon FPS sandbox endpoints in a library-dependant way. For example, to use the C# library, you set your security credentials in the `[package-root]/src/Amazon.FPS.Samples/Amazon.FPS.Samples/AmazonFPSSamples.cs` file, while for the perl library you set them in the individual `[package-root]/src/Amazon/FPS/Samples/*.pl` file you are working with.

In the following section, we show how to work with the `VerifySignature` sample using the Java library. You will use this fundamental API frequently for server-side validation of your return URL responses and IPN notifications. You will find that the basic process you use for the `VerifySignature` sample is the same for all the other samples in the `FPS/Samples` (or, in the case of `Amazon.FPS.Samples`) folder. (The process for the CBU and Return URL/IPN Validations samples are different. For more information, see [Understanding the Amazon CBU Samples \(p. 129\)](#) and [Understanding the IPNAndReturnURLValidation Sample \(p. 133\)](#).)

## Understanding the VerifySignature Sample

This section explains how to use the Java version of the `VerifySignature` API. If you want to use one of the other sample libraries, they are set up nearly identical to the Java sample. To see file locations for the `VerifySignature` sample for your preferred language, see [Locations of the VerifySignatureSample Files in Other Libraries \(p. 128\)](#).

To use the sample, do the following

### Using the VerifySignature Sample

1	Set up your programming environment so that the program will compile without warnings or errors. For the Java sample, this includes ensuring that the files and sub folders in the <code>[package-root]/src</code> and <code>[package-root]/third-party</code> folder are in the java classpath.
---	--

2	<p>In the <code>[package-root]/src/config.properties</code> file, set the values for <code>AwsAccessKey</code> and <code>AwsSecretKey</code> using your security credentials. To get your credentials, see <i>Getting an AWS Account</i> in the <a href="http://docs.amazonwebservices.com/AmazonFPS/latest/FPSGettingStartedGuide/">http://docs.amazonwebservices.com/AmazonFPS/latest/FPSGettingStartedGuide/</a>.</p> <p>In the same file, if you want to target the sandbox, change the <code>AwsServiceEndPoint</code> property to <a href="http://fps.sandbox.amazon.com">http://fps.sandbox.amazon.com</a>. Then save the file.</p>
3	<p>In the <code>[package-root]/src/com/amazonaws/fps/samples/VerifySignature.java</code> file, find the section containing the lines:</p> <pre>VerifySignatureRequest fpsRequest = new VerifySignatureRequest();     // @TODO: set request parameters here     // invokeVerifySignature(service, fpsRequest);</pre> <p>(The <code>VerifySignatureRequest</code>, <code>VerifySignatureResult</code>, and <code>VerifySignatureResponse</code> classes are located in <code>[package-root]/src/com/amazonaws/fps/model</code> folder.)</p>
4	<p>In the same file, remove the comment on <code>invokeVerifySignature</code>, and after it add the <code>VerifySignature</code> parameter assignments consistent with your transaction. For example:</p> <pre>fpsRequest.setAction("VerifySignature"); fpsRequest.setUrlEndpoint("http://myApplication/my-ipn-response.pgp"); fpsRequest.setHttpParameters(     "Name1=Joe&amp;     "Name2=College&amp;" +     "signatureVersion=2&amp;" +     "signatureMethod=HMACSHA256&amp;" +     "certificateUrl=https://fps.amazonaws.com/cert/key.pem&amp;" +     "signature=aoeuAOE123eAUDhf");</pre> <p>Save the file. For information on the parameters to <code>VerifySignature</code>, see <a href="#">VerifySignature (p. 93)</a>.</p>
5	<p>Compile and run the sample. The program copies to standard out a representation of the <code>VerifySignatureResponse</code> XML fragment similar to the following:</p> <pre>VerifySignature Action Response ===== VerifySignatureResponse VerifySignatureResult   True VerificationStatus   Success ResponseMetadata   RequestId     bda6-4f5f-b37b-1a146b9a-b9e45c3012a5:0</pre> <p>For information on the XML document returned by <code>VerifySignature</code>, see <a href="#">VerifySignature (p. 93)</a>.</p>

In addition to simple API invocation, the samples provide you the following advanced options:


- The ability to simulate a mock Amazon FPS service and get responses without a live connection.

- Specifying a proxy host and port, through `config.properties`.
- Setting the endpoint, through `config.properties`
- Logging, through `log4j.properties`

## Locations of the VerifySignatureSample Files in Other Libraries

The development libraries for C#, Perl, and PHP also enable you to perform a server-side validation of a signature in a return URL or IPN notification. The following tables list the locations of the files referenced in [Understanding the Amazon FPS Samples \(p. 126\)](#).

### C# File Locations for the Amazon.FPS VerifySignature Sample

File	Location
<code>VerifySignatureRequest.cs</code>	<code>[package root]/src/Amazon.FPS/Amazon.FPS.Model</code>
<code>VerifySignatureSample.cs</code>	<code>[package root]/src/Amazon.FPS.Samples/Amazon.FPS.Samples</code>
<code>Amazon.FPS.proj</code> (Visual Studio.NET project for the FPS development library)	<code>[package root]/src/Amazon.FPS/Amazon.FPS</code>
<code>Amazon.FPS.Samples.proj</code> (Visual Studio.NET project for the Amazon FPS API samples)	<code>[package root]/src/Amazon.FPS/Amazon.FPS.Samples/Amazon.FPS.Samples</code>
<code>Amazon.FPS.sln</code> Visual Studio.NET solution for the library package	<code>[package root]/src/Amazon.FPS/Amazon.FPS</code>
 <b>Note</b>  The Visual Studio.NET samples are organized into this solution. After setting your access parameters the first time, you build the entire solution to generate the dependency classes. Then you modify the specific sample you want. See the <code>Readme.html</code> file for more information.	

### Perl File Locations for the Amazon.FPS VerifySignature Sample

Class	Location
<code>VerifySignatureRequest.pm</code>	<code>[package root]/src/Amazon/FPS/Model</code>
<code>VerifySignatureSample.pl</code>	<code>[package root]/src/Amazon/FPS/Samples</code>
<code>ReadMe.html</code> (readme for perl fps library)	<code>[package root]/src</code>

### PHP File Locations for the Amazon.FPS VerifySignature Sample

Class	Location
VerifySignatureRequest.php	<code>[package root]/src/Amazon/FPS/Model</code>
VerifySignatureSample.php	<code>[package root]/src/Amazon/FPS/Samples</code>
ReadMe.html (readme for php fps library)	<code>[package root]/src</code>

## Understanding the Amazon CBUI Samples

Amazon provides five samples in four programming languages (C#, Java, Perl, and PHP) which show you how to build Co-Branded User Interface request URLs.

When you download a sample file, such as `amazon-fps-2008-09-17-java-library`, the `[package root]/src/com/amazonaws/cbui/samples` folder contains sample classes showing how to generate pipeline-specific CBUI URLs from your code (`[package root]` is the location you extracted your sample package).

Each sample describes its requirements in its `Readme.html` file, located at the package root. Typically, the entire library structure must be available to the compiler. For example, the `amazon-fps-2008-09-17-php-library.zip` file contains the `src/Amazon/CBUI` and folders which the files in `src/Amazon/CBUI/Samples` require.

In addition to these primary components, a sample may include other required resources. For example, the Java samples all include numerous jar files in the `[package-root]/third-party` folder, which must also be in your classpath in order to compile the sample.

For each sample, you must set your security credentials and Amazon FPS sandbox endpoints in a library-dependant way. For example, to use the C# library, you set your security credentials in the `[package-root]/src/Amazon.FPS.Samples/Amazon.FPS.Samples/AmazonFPSSamples.cs` file, while for the perl library you set them in the individual `[package-root]/src/Amazon/CBUI/Samples/*.pl` file you are working with.

The following samples are provided with each sample library:

Class	Description
<code>CBUIEditTokenPipeline Sample</code>	Requests authorization for a one-time payment.
<code>CBUIMultitUsePipeline Sample</code>	Requests authorization for multiple payments.
<code>CBUIRecipientPipeline Sample</code>	Requests authorization for a recipient token pipeline, such as that needed for marketplace fixed and variable fees.
<code>CBUIRecurringTokenPipeline Sample</code>	Requests authorization for a recurring token pipeline, such as that needed for periodic charges.
<code>CBUISingleUsePipelineSample</code>	Requests authorization for an edit-token pipeline.

In the following section, we show how to work with the `CBUISingleUsePipeline` sample using the Java library. This sample enables you to set up a single-use token for a one-time payment.

You will find that the basic process you use for the `CBUISingleUsePipeline` sample is the same for all the other samples in the `CBUI/Samples` (or, in the case of `Amazon.CBUI.Samples`) folder. (The process for the FPS and Return URL/IPN Validations samples are different. For more information, see [Understanding the Amazon FPS Samples \(p. 126\)](#) and [Understanding the IPNAndReturnURLValidation Sample \(p. 133\).](#))

## Java

This section describes the Java version of the `CBUISingleUsePipeline`. The files for the C#, Perl, and PHP `CBUISingleUsePipeline` samples are listed in [Locations of the CBUISingleUsePipeline Files in Other Libraries \(p. 131\)](#)

The `CBUISingleUsePipeline` sample centers on the following files:

File	Description
<code>config.properties</code>	Set your AWS access key ID, AWS secret key, and sandbox endpoint in this file.
<code>CBUISingleUsePipelineSample.java</code>	In the main method, you create an <code>AmazonFPSSingleUsePipeline</code> object and use it to add parameters specific to your application.
<code>AmazonFPSSingleUsePipeline.java</code>	Invoked from <code>CBUISingleUsePipelineSample.java</code> , this class contains the <code>setMandatoryParameters</code> and <code>validateParameters</code> functions which you can customize for your application.

### Co-Branded service request with Java SDK Sample

1	Open the file <code>[package-root]/src/config.properties</code> , and set <code>AwsAccessKey</code> and <code>AwsSecretKey</code> properties to your AWS access key and AWS secret key, respectively. To get your security credentials, see <a href="#">Getting an AWS Account</a> in the <a href="#">Amazon Flexible Payments Service Getting Started Guide</a> .
2	In the same file, set the <code>AwsServiceEndPoint</code> to <code>https://fps.sandbox.amazonaws.com/</code> (the Amazon FPS sandbox).
3	In the same file, set the <code>CBUIServiceEndPoint</code> to <code>https://authorize.payments-sandbox.amazon.com/cobranded-ui/actions/start</code> (the Co-Branded service sandbox).

**Amazon FPS Advanced Quick Start Developer Guide**  
**Locations of the CBUISingleUsePipeline**  
**Files in Other Libraries**

4	<p>Open the file <code>[package-root]/src/com/amazonaws/cbui/samples/CBUISingleUsePipelineSample.java</code>, and find the following line:</p> <pre>AmazonFPSSingleUsePipeline pipeline= new AmazonFPSSingleUsePipeline(accessKey, secretKey);</pre> <p>Change the <code>pipeline.setMandatoryParameters</code> and <code>pipeline.addParameters</code> method calls to the following:</p> <pre>//pipeline name, your return URL, and the amount pipeline.setMandatoryParameters("callerReferenceSingleUse", "[your returnUrl]", "5");  //optional parameters pipeline.addParameter("currencyCode", "USD"); pipeline.addParameter("paymentReason", "Now and Forever - Richard Mark"); pipeline.addParameter("paymentMethod", "ABT,ACH,CC"&gt;; pipeline.addParameter("callerReference", "[Unique ID for the transaction]");</pre> <p>Save the file.</p>
5	Ensure that all the jar files in the <code>third-party</code> folder and sub folders are in your java CLASSPATH.
6	Compile and run the sample. The Co-branded authorization page is printed to standard out.
7	Using a web browser, navigate to the URL produced by the sample. You must use an account different from your AWS developer or business accounts.
8	When complete, the page you specified as <code>[your returnUrl]</code> is hit with the Co-Branded service response. If signed, you validate this response by testing the signature. For information on validating the signature, see <a href="#">Understanding the IPNAndReturnURLValidation Sample (p. 133)</a> .

You can customize the sample by modifying the file `[package-root]/src/com/amazonaws/cbui/AmazonFPSSingleUsePipeline.java`. The `setMandatoryParameters` only requires `callerReference`, `returnUrl`, and `transactionAmount`. If you want to make more parameters mandatory, modify this method.


In the same file, the `validateParameters` function ensures that the `transactionAmount` parameter is present. You can add custom validation checks to this method.

## Locations of the CBUISingleUsePipeline Files in Other Libraries

The development libraries for C#, Perl, and PHP also enable you to create CBUI pipeline urls. The following tables indicate the locations of the files referenced in [Understanding the Amazon CBUI Samples](#).

**Amazon FPS Advanced Quick Start Developer Guide**  
**Locations of the CBUI SingleUsePipeline**  
**Files in Other Libraries**

**C# File Locations for the Amazon.FPS CBUI Sample**

File	Location
CBUISingleUsePipelineSample.cs	<i>[package root]</i> \src\Amazon.CBUI \Amazon.CBUI.Model.
AmazonFPSSingleUsePipeline.cs	<i>[package root]</i> \src \Amazon.CBUI.Samples \Amazon.CBUI.Samples
Amazon.CBUI.proj (Visual Studio.NET solution for the development library)	<i>[package root]</i> \src\Amazon.CBUI \Amazon.CBUI\
Amazon.CBUI.Samples.proj (Visual Studio.NET solution for the Amazon FPS API samples)	<i>[package root]</i> \src\Amazon.CBUI \Amazon.CBUI.Samples \Amazon.CBUI.Samples
Amazon.FPS.sln Visual Studio.NET solution for the library package	<i>[package root]</i> /src/Amazon.FPS/ Amazon.FPS
 <b>Note</b>  The Visual Studio.NET samples are organized into this solution. After setting your access parameters the first time, you build the entire solution to generate the dependency classes. Then you modify the specific sample you want. See the <code>Readme.html</code> file for more information.	

**Perl File Locations for the Amazon.FPS VerifySignature Sample**

Class	Location
CBUISingleUsePipelineSample.pl	<i>[package root]</i> /src/Amazon/CBUI/ Samples.
AmazonFPSSingleUsePipeline.pm	<i>[package root]</i> /src/Amazon/CBUI
ReadMe.html (readme for perl fps library)	<i>[package root]</i> /src

**PHP File Locations for the Amazon.FPS VerifySignature Sample**

Class	Location
CBUISingleUsePipelineSample.php	<i>[package root]</i> /src/Amazon/CBUI/ Model.
CBUISingleUsePipeline.php	<i>[package root]</i> /src/Amazon/CBUI/ Samples
ReadMe.html (readme for php fps library)	<i>[package root]</i> /src



## Understanding the IPNAndReturnURLValidation Sample

Amazon provides samples in four programming languages which show you how to perform a client-side verification of the signatures in both the return URL and in IPN notifications. In this section, we will briefly go over the essential details of the Java version only. The other samples differ only in the programming language used for rendering them. For specific comprehensive information on a particular sample, see its `IPNAndReturnURLValidation.html` file.



### Note

For most applications, Amazon recommends server-side signature verification using `VerifySignature`. For more information, see [VerifySignature \(p. 93\)](#)

Each `IPNAndReturnURLValidation` sample contains three primary components in the `src/com/amazonaws/ipnreturnurlvalidation` folder. These are:

File	Description
<code>ReturnUrlVerificationSampleCode.java</code>	This class contains the program entry point for verifying the signature contained in a return URL, and thereby validating the return URL content. It sets up initial parameter values for return URL responses, and then calls the static method <code>SignatureUtilsForOutbound.validateRequest</code> with those values.
<code>IPNVerificationSampleCode.java</code>	This class contains the program entry point for verifying the signature contained in an IPN notification. It sets up initial parameter values for IPN notifications, and then calls the static method <code>SignatureUtilsForOutbound.validateRequest</code> with those values.
<code>SignatureUtilsForOutbound.java</code>	Invoked from <code>ReturnUrlVerificationSampleCode.java</code> and <code>IPNVerificationSampleCode.java</code> , this class uses the signature version 2 process to validate the signature contained in the <code>certificateUrl</code> parameter using PKI. It contains methods to reassemble the string to sign, URL encode the string, and sign it using the Amazon certificate listed as the signer. Finally, it validates the signature and prints the result to standard out.

In addition to these primary components, a sample may include other required resources. For example, the Java samples all include the `third-party` folder, the jar files of which must be in your classpath in order to compile the sample.

To use the sample, do the following

### Using the Standard Button IPNAndReturnURLValidation Sample for PKI-based Validation

1	Set up your programming environment so that the program will compile without warnings or errors. For the Java sample, this includes ensuring that the <code>src/com/amazonaws/ipnreturnurlvalidation</code> folder and the files are available to the compiler, either by including them as command line parameters, or, if you build using an IDE, by including them as project resources.
2	The <code>ReturnUrlVerificationSampleCode</code> and <code>IPNVerificationSampleCode</code> classes use a <code>HashMap</code> to store parameters which correspond to the fields returned during a return URL response or an IPN notification. Modify these values to suit the response you want to validate. These are the only values you need to change using this sample.
4	<p>Compile the sample. For example, if you are including the <code>[package-root]src/third-party/commons-codec-1.3/commons-codec-1.3.jar</code> using the linux command line, you would type</p> <pre>\$javac -cp .:[package-root]src/third-party/commons-codec-1.3/commons-codec-1.3.jar ReturnUrlVerificationSampleCode.java SignatureUtilsForOutbound.java</pre> <p>On Windows, you would type</p> <pre>\$javac -cp .:[package-root]src/third-party/commons-codec-1.3/commons-codec-1.3.jar SignatureUtilsForOutbound.java</pre>
5	<p>Run the sample. Continuing the previous example, on linux, you would type</p> <pre>\$javac -cp .:[package-root]src/third-party/commons-codec-1.3/commons-codec-1.3.jar ReturnUrlVerificationSampleCode</pre> <p>On Windows, you would type</p> <pre>\$javac -cp .:[package-root]src/third-party/commons-codec-1.3/commons-codec-1.3.jar ReturnUrlVerificationSampleCode</pre> <p>The result <b>"Is signature correct: true"</b> is printed to standard out if the verification determines the signature to be valid.</p>

## Locations of the IPNAndReturnURLValidation Files in Other SDKs

The development libraries for C#, Perl, and PHP also enable you to test Return URL and IPN notifications. The following tables indicate the locations of the files referenced in [Understanding the IPNAndReturnURLValidation Sample](#).

### C# File Locations for the Amazon.IpnReturnUrlValidationSample Library

File	Location
<code>ReturnUrlVerificationSampleCode.cs</code>	<code>[package root]src\Amazon.IpnReturnUrlValidation\.</code>
<code>IPNVerificationSampleCode.cs</code>	<code>[package root]src\Amazon.IpnReturnUrlValidationSamples\IpnReturnUrlValidationSamples\</code>
<code>SignatureUtilsForOutbound.cs</code>	<code>[package root]src\Amazon.IpnReturnUrlValidationSamples\IpnReturnUrlValidationSamples\</code>

File	Location
IpnAndReturnUrlValidation.html (readme for this sample)	<code>[package root]src\Amazon.IpnReturnUrlValidationSamples.IpnReturnUrlValidationSamples\</code>
IpnReturnUrlValidation.Samples.csproj (Visual Studio.NET project for this sample)	<code>[package root]src\Amazon.IpnReturnUrlValidation\</code>

### Perl File Locations for the IpnReturnUrlValidation Library

Class	Location
ReturnUrlVerificationSampleCode.pl	<code>[package root]src/Amazon/IpnReturnUrlValidation/Samples.</code>
IPNVerificationSampleCode.pl	<code>[package root]src/Amazon/IpnReturnUrlValidation/Samples</code>
SignatureUtilsForOutbound.pm	<code>[package root]src/Amazon/IpnReturnUrlValidation</code>
IpnAndReturnUrlValidation.html (readme for this sample)	<code>[package root]src</code>

### PHP File Locations for the IpnReturnUrlValidation Library

Class	Location
ReturnUrlVerificationSampleCode.php	<code>[package root]src/Amazon/IpnReturnUrlValidation/Samples.</code>
IPNVerificationSampleCode.php	<code>[package root]src/Amazon/IpnReturnUrlValidation/Samples</code>
SignatureUtilsForOutbound.php	<code>[package root]src/Amazon/IpnReturnUrlValidation</code>
IpnAndReturnUrlValidation.html (readme for this sample)	<code>[package root]src</code>

## Getting the Samples

The Amazon FPS sample applications are available from the Amazon Web Services developer center.

### To download Amazon FPS samples:

1. Go to <http://developer.amazonwebservices.com/connect/forumindex.jspa>.

The **Discussion Forums** page opens.

2. From the **Resources** menu, choose **Sample Code & Libraries**.
3. In the **Browse by Category** area, choose **Amazon Flexible Payments Service**.
4. Choose your sample of interest in the programming language you prefer. To obtain the sample applications listed in this guide, look for sample applications whose package name resembles the

format "amazon-fps-2008-09-17-*LANGUAGE*-library". For example, the Java sample is available in the file **amazon-fps-2008-09-17-java-library.zip**.

5. Read the instructions on the page. Note that this page enables you to start a community discussion about sample. You can also review it. When you are ready to proceed, click **Download**.

The **Opening Amazon** window opens. Ensure it is the sample you want, and Click **OK**

6. Extract the zipped files to a convenient location on your workstation.

Each download includes sample-specific instructions in its README.txt file. For general guidance on the samples applicable to this edition of Amazon FPS , see [Code Samples \(p. 125\)](#).

# Appendix: Verifying Responses Signed Using Signature Version 1

---

If the *SignatureVersion* parameter of your Return URL or IPN response has a value of *1*, and you want to validate the response, you must use the following process to verify the legitimacy of a return from Amazon Payments.



## Important

The previous method for signing will expire on 01 November, 2010. At that time, any signing you do with your access keys must be done using the new method.

Because of the impending expiration of signature version 1, you should migrate to signature version 2 as soon as you can. For more information, see [Appendix: Moving your Application to Signature Version 2](#) (p. 139)

## Verifying a return signature

1	Decrypt the request.
2	Read the AWS Access Key ID from the request. Check that it is a valid Access Key ID.
3	Use the AWS Access Key ID value to look up the value of your Secret Key.
4	Remove the signature parameter and its value from the request.
5	Use your Secret Key and the remainder of the request to compute the signature of the request. To generate a signature, see <a href="#">Working with Signatures</a> (p. 41)
6	Compare that signature with the signature in the original request.
7	The signatures must match. If they do not, an error is returned.

## Access Key Rotation Considerations with Signature Version 1

If you enable [access key rotation](#) using signature version 1, the [outbound notifications](#) will be signed according to the rules in the following table. These rules are dependant on three conditions:

**Amazon FPS Advanced Quick Start Developer Guide**  
**Access Key Rotation Considerations**  
**with Signature Version 1**

---

- Your account has two active key pairs (referred to below as *K1* and *K2*)
  - *K1* was created before *K2*.
  - You use *K2* to sign the incoming request (FPS API/CBUI pipeline/Simple Pay button request)
1. The responses will be signed using *K2* for all the corresponding outbound notifications generated by the request.
  2. If you deactivated *K2* before all the outbound notifications are generated, the signatures for all pending notifications will be generated using *K1* (the oldest active key). For example:
    3. a. A `Pay` request is signed using *K2*.
    - b. The `Payment Initiated` IPN is signed using *K2* and sent to the specified IPN endpoint.
    - c. *K2* is deactivated or deleted and a new key, *K3*, is created.
    - d. The `Payment Successful` IPN is signed using *K1* and sent to the specified IPN endpoint.
  4. If all the keys are deactivated or deleted before the outbound notification is generated, we send the notification without any signature.

For information about [access key rotation](#), see [Access Key Rotation \(p. 45\)](#).

# Appendix: Moving your Application to Signature Version 2

---

Signature version 1 expires on 01 November, 2010. After that date, applications using signature version 1 will not produce correct results. This sections details the process you use to migrate to signature version 2 for [inbound requests](#) and *outbound responses*.

Amazon provides code samples to assist you in migrating your signature version 1 code to signature version 2.

Language	File
C#	<a href="#">SigV2_MigrationSampleCode_CS.zip</a>
Java	<a href="#">SigV2_MigrationSampleCode_Java.zip</a>
Perl	<a href="#">SigV2_MigrationSampleCode_Perl.zip</a>
PHP	<a href="#">SigV2_MigrationSampleCode_PHP.zip</a>
Ruby	<a href="#">SigV2_MigrationSampleCode_Ruby.zip</a>

The following sections describe the high-level process for migrating your signature version 1 signing code.

## Migrating Inbound Requests to Signature Version 2

If you are currently signing your buttons using a signature calculated using signature version 1, and want to begin using signature version 2, you need to make the following changes:

- Modify the way you assemble your signature. For more information, see [Working with Signatures \(p. 41\)](#) and [Differences Between Signing Versions \(p. 140\)](#).

- Include the value `2` for the required `SignatureVersion` parameter in your button forms and FPS actions.
- Determine your preferred signing algorithm (either the preferred `HmacSHA256`, or `HmacSHA1`), and set the `SignatureMethod` parameter to the corresponding value in your button forms and FPS actions.
- Check that your encryption method supports your chosen algorithm, and rewrite it if necessary.

After you change your code, validate it in the sandbox. You can check your signature code using the samples listed in this guide. For more information, see [Working with Signatures \(p. 41\)](#).

## Migrating Outbound Notifications to Signature Version 2

If you are currently validating return URL and IPN notifications using signatures build using signature version 1, and want to begin using signature version 2, you need to do the following:

Adjust the way you build the signature to comply with signature version 2. For more information, see [Working with Signatures \(p. 41\)](#) and [Differences Between Signing Versions \(p. 140\)](#).

- Enable signature version 2 in your account settings. For more information, see [Amazon Flexible Payments Service Getting Started Guide](#).
- Replace your core validation code with a call to the FPS action, [VerifySignature \(p. 93\)](#). This server side solution is the method we recommend.
- Or, if your application's performance requires it, you can replace your core validation code with a [client-side signature verification](#) method using the value returned in the `certificateUrl` parameter. For more information, see [Client-side Signature Validation \(p. 44\)](#). For an example of how to do this in code, see [Understanding the IPNAndReturnURLValidation Sample \(p. 133\)](#).

## Differences Between Signing Versions

As of 01 November, 2010, the signature version 1 is no longer supported. While the security benefits of the new version are significant, the implementation differences between the two are few:

- You create the concatenated URL string differently:

Include additional components, including null parameters

Include the query string control characters '=' and '&'

Sort the query string parameters using byte ordering

URL-encode the concatenated URL string before signing

- You can now use HMAC-256 for signing [inbound requests](#). Although we prefer HMAC-256, HMAC-SHA1 is also supported. For outbound notifications, we support the RSA-SHA1 algorithm.
- You use the new `signatureMethod` parameter to indicate the signing algorithm (valid values are `HmacSHA256` or `HmacSHA1`).
- You include the new `signatureVersion` parameter, which must be set to `2`.



# Signature Version 2 FAQ

---

## **General Questions**

1. *Why is Amazon Payments upgrading from signature version 1 to 2?*

The new signature protocol enhances the security for inbound API requests and outbound notifications. Amazon Web Services has already released signature version 2 for Amazon EC2, Amazon Simple DB, and Amazon SQS, and has deprecated signature version 1 for these services. Additional information on signature version 2 can be found here <http://www.amazon.com/gp/blog/post/PLNK14GNSFUFPRSOA> and <http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1928>.

2. *Is there a timeframe to deprecate signature version 1 for FPS and Simple Pay?*

Yes. Signature version 1 will be deprecated on 1 November 2010. After that date you must use signature version 2 for signing your requests and verifying responses.

3. *What do I need to do to migrate to signature version 2?*

- You need to migrate all your inbound requests (FPS API requests, CBUI/Simple Pay requests) to use signature version 2.
- You need to start verifying signatures in outbound notifications (CBUI/Simple Pay return URLs and IPNs) using signature version 2.

The exact steps to migrate from version 1 to 2 are documented in all of our [FPS Quick Start guides](#) and the Amazon Simple Pay [Advanced User Guide](#). You can also download migration samples that can assist you in the language of your choice: [Java](#) | [C#](#) | [PHP](#) | [Perl](#)

4. *What changes can we expect going forward?*

- Amazon Flexible Payments Service, Amazon Payments Cobranded UI and Amazon Simple Pay will start supporting both signature version 1 and version 2 for inbound API requests.
- Amazon Payments will support the option to sign outbound notifications (Return URLs and IPNs) using both signature version 1 and 2.

## **Differences between Signature Version 1 and Signature Version 2**

1. *How is the new signature algorithm different from the existing one for inbound API requests?*

If you're familiar with signature version 1, these are the main differences with signature version 2:

- The string to sign is formed in a different way.

- HMAC-SHA256 can be used when the request is signed (we prefer HMAC-SHA256, but we still support HMAC-SHA1).
  - The *SignatureVersion* request parameter should be set to 2.
  - The *SignatureMethod* request parameter should be set to either *HmacSHA256* or *HmacSHA1* to indicate which signing method you want.
2. *How is the new signature algorithm different from the existing one for outbound notifications (CBUI\Amazon Simple Pay return URLs and IPNs)?*

If you're familiar with validating signature using version 1, these are the main differences with validating signature using version 2:

- The string to sign is formed in a different way.
  - The signature computed is based on asymmetric keys (PKI and RSA-SHA1 encryption) instead of symmetric (using the developer's AWS Access Key ID and AWS Secret Key).
  - Signature validation is simpler do to a *VerifySignature* API that can validate the signature returned in IPNs and Return URLs.
3. *What is different about forming the string to sign for signature version 2 (inbound and outbound)?*
- Additional components of the request are included in the string to sign (HTTP method – GET \POST, Server URL endpoint).
  - The query string control parameters (the equal sign and ampersand) are included in the string to sign.
  - The query string parameters are sorted using byte ordering.
  - The query string parameters and their values are URL encoded before signing the request.
4. *Why does Amazon Web Services use asymmetric instead of symmetric keys to compute the signature for outbound notifications?*

Signature computation for outbound notifications on the AWS end and signature validation on your end using symmetric keys becomes cumbersome and indeterministic by the introduction of AWS access key rotation (refer to <https://aws-portal.amazon.com/gp/aws/developer/account/index.html?ie=UTF8&action=access-key> for more details). Asymmetric key based signing will become an AWS wide standard for outbound notifications.

### ***Options to validate the signature in outbound notifications (CBUI\Simple Pay return URLs and IPNs)***

1. *What are the options available to validate the signature?*

The signature version 2 security has two methods for you to verify the signature of the responses from Amazon Payments

- Server-side signature verification using the *VerifySignature* API
  - Client-side signature verification using PKI
2. *Who should use *VerifySignature* API?*
- Developers/merchants who want to quickly integrate with our service.
  - Small developers/merchants who do not worry about the latency introduced by an extra API call and want to keep their code simple by executing a remote API call to verify the signature.
  - Amazon Simple Pay merchants who do not have technical expertise to implement signature validation.
3. *Who should validate the signature on their end using PKI?*
- Medium to large developers/merchants who worry about scaling and performance because of the network latency introduced by making additional API calls to verify the signature.
  - Developers and merchants who require a fallback mechanism for the *VerifySignature* API.

### **About Verify Signature API**

#### 1. How does the *VerifySignature* API work?

The *VerifySignature* API is pretty straightforward to use. Send the entire URL with the HTTP parameters received to the *VerifySignature* API and it returns a Boolean expression that indicates whether the signature was validated or not.

#### 2. How do I access the *VerifySignature* API?

The API is hosted on the FPS endpoint along with all the other APIs:

- <https://fps.amazonaws.com/> - Production
- <https://fps.sandbox.amazonaws.com/> - Sandbox

#### 3. Is the *VerifySignature* API different from the other APIs exposed by FPS?

Yes it is.

- This API does not require any authentication/authorization. Since the API is hosted at a HTTPS endpoint, the data sent is secure and confidential.
- This API also does not require you to sign up for a developer account.

### **About PKI-based Signature Validation**

#### 1. About PKI based signature validation

Though we recommend the *VerifySignature* API, if you are worried about an additional API call, the following steps detail the signature validation procedure:

- Decode the signature in the notification.
- Decode and read the *signatureVersion* and *signatureMethod* parameters from the notification. The value of *signatureVersion* value should be 2, and the value for *signatureMethod* value should be *RSA-SHA1* (format is Algorithm-Digest).
- Decode and read the *certificateUrl* parameter from the notification.
- Verify that certificate corresponding to the URL was already downloaded and cached.
- If the certificate was not cached, download and cache it.
- Compute the *StringToSign* parameter using the signature version 2 algorithm (same as [inbound requests](#)). *StringToSign* should include all the parameters sent in the notification excluding the signature.
- Calculate the signature using the PKI based cryptography, using the *StringToSign* value you created in the previous step and the certificate you cached earlier.
- Compare the calculated signature you just calculated with the signature you received in the original notification.
- If the signatures match, process the notification. Otherwise, discard the notification.

#### 2. How can I get the certificate?

The certificate is hosted on a trusted location and the URL to download the certificate will be provided along with the [outbound notifications](#).

#### 3. Why does AWS provide the path to the certificate instead of the entire certificate in [outbound notifications](#)?

When notification happens through the browser, the length of the notification for return URLs is restricted by the maximum URL length supported by the browser (2083 characters in IE - <http://support.microsoft.com/kb/208427>)

#### 4. Should I validate the certificate before using it to verify the signature?

No, you don't need to validate the certificate, since it will be hosted in a trusted location under the control of Amazon FPS. Since the cumbersome step of certificate validation is eliminated, verifying signatures using asymmetric keys becomes as simple as using symmetric keys.

5. *How often should I download the certificate?*

We recommend you to cache the certificate using a certificate URL as the key so that you can prevent downloading it every time you want to validate the signature. When the certificate expires (once every year), we will host a new certificate in an alternate URL and start signing the notifications using the new one. Since the certificate URL in the notification changes, your caching implementation will automatically download the certificate again, cache the new one, and start using it for signature validation.

### **Questions/Support**

1. *I am an existing customer. How can I get help with migration?*

The exact steps to migrate from version 1 to 2 are documented in all of our [FPS Quick Start guides](#) and the Amazon Simple Pay Advanced User Guide. You can also download migration samples that can assist you in the language of your choice: [Java](#) | [C#](#) | [PHP](#) | [Perl](#)

If you face any problems, please feel free to contact-us on the [Amazon FPS Developer Forum](#).

2. *I have concerns about migrating to signature version 2. What do I do?*

Please go to <http://aws.amazon.com/contact-us> to file a Contact-Us request. We will address your concerns in the best possible way.

# Glossary

---

access key rotation	For added security, you can switch between an active and inactive access key on your AWS security credentials page.
AWS Access Key ID	A string distributed by AWS that uniquely identifies your AWS developer account. You include this ID in every request.
buyer	Customer making a purchase. Also called a sender.
caller	A developer who facilitates payment between a sender and a recipient.
client-side signature verification	With Amazon FPS, you can use client-side PKI-based verification to validate IPN and Return URL responses with the value of the <i>certificateUrl</i> parameter. The <i>certificateUrl</i> parameter is provided in each Return URL and IPN response.
endpoint	The URI that specifies the destination of an API request.
HMAC	The Hash Message Authentication Code used to authenticate a message. The HMAC is calculated using a standard, hash cryptographic algorithm, such as SHA-256. This algorithm uses a key value to perform the encryption. That key is your <a href="#">Secret Key</a> . For that reason, your Secret Key must remain a shared secret between you and Amazon Payments.
inbound requests	Button click or other form request to Amazon Payments. Also inbound notification.
Instant Payment Notification	A notification that is sent whenever a payment, refund, or reserved payment completes successfully or fails. The caller must host this notification service and provide Amazon Payments with its URL.
marketplace	An environment in which the caller charges a fee for facilitating a transaction between a sender and a recipient.
order pipeline	The steps through which an order passes between the time a customer selects an item and the customer's pay instrument is charged.

outbound notifications	Response from Amazon Payments to your Amazon FPS application by way of Return URL or IPN.
PKI	, Public Key Infrastructure. A set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates. Amazon FPS uses PKI in the Return URL and IPN notifications it sends to your application in response to an FPS action, rest request or query o.
recipient	A seller who receives a payment from a buyer (sender) in exchange for a service or product.
string-to-sign	Prior to calculating the HMAC signature, you first assemble the components for the signature in a sorted order, and then URL encode them. The pre-encrypted string is the string-to-sign.
reserve	The amount that is put in reserve against a credit card but not charged. Later, the transaction is settled (typically when the product is actually shipped).
sandbox	A part of the Amazon Payments web service where you can test the functionality of your application without incurring charges or purchasing products.
Secret Key	A string distributed by AWS that uniquely identifies your AWS developer account. The Secret Key is a shared secret between the developer and AWS. The Secret Key is used as the key in the HMAC algorithm that encrypts the signature.
seller	Same as a recipient.
sender	The sender (also known as the buyer) pays a recipient for a product or service.
settle	To complete a transaction that has been reserved. If you don't charge the sender immediately upon the initiation of the purchase (and instead reserve the amount against the sender's credit card), you settle the transaction later, typically after you ship the product to the sender. Settle actually makes the reserved amount move from the sender to the recipient.
SHA1, SHA256	Secure Hash Algorithms used for Amazon Web Services signatures. SHA1 is an earlier version of the algorithm, which is currently being deprecated for Amazon Web Services. SHA256 is its more secure replacement.
signature	A URL-encoded string composed of request parameters and their values encrypted using an HMAC algorithm. Signatures are used to authenticate and safeguard requests.


# Document Conventions

---

This section lists the common typographical and symbol use conventions for AWS technical publications.

## Typographical Conventions

This section describes common typographical use conventions.

Convention	Description/Example
Call-outs	A call-out is a number in the body text to give you a visual reference.  The reference point is for further discussion elsewhere.
Code in text	Inline code samples (including XML) and commands are identified with a special font. You can use the command <code>java -version</code> .
Code blocks	Blocks of sample code are set apart from the body and marked accordingly. <pre># ls -l /var/www/html/index.html -rw-rw-r-- 1 root root 1872 Jun 21 09:33 /var/www/html/ index.html # date Wed Jun 21 09:33:42 EDT 2006</pre>
Emphasis	Unusual or important words and phrases are marked with a special font. You <i>must</i> sign up for an account before you can use the service.
Internal cross references	References to a section in the same document are marked. For more information, see <a href="#">Document Conventions (p. 147)</a> .
Logical values, constants, and regular expressions, abstracta	A special font is used for expressions that are important to identify, but are not code. If the value is <code>null</code> , the returned response will be <code>false</code> .
Product and feature names	Named AWS products and features are identified on first use. Create an <i>Amazon Machine Image</i> (AMI).

Convention	Description/Example
Operations	In-text references to operations. Use the <code>GetHITResponse</code> operation.
Parameters	In-text references to parameters. The operation accepts the parameter <code>AccountID</code> .
Response elements	In-text references to responses. A container for one <code>CollectionParent</code> and one or more <code>CollectionItems</code> .
Technical publication references	References to other AWS publications. If the reference is hyperlinked, it is also underscored. For detailed conceptual information, refer to the <i>Amazon Mechanical Turk Developer Guide</i> .
User entered values	A special font marks text that the user types. At the password prompt, type <b>MyPassword</b> .
User interface controls and labels	Denotes named items on the UI for easy identification. On the <b>File</b> menu, click <b>Properties</b> .
Variables	When you see this style, you must change the value of the content when you copy the text of a sample to a command line. <code>% ec2-register &lt;your-s3-bucket&gt;/image.manifest</code> See also the following symbol convention.

## Symbol Conventions

This section describes the common use of symbols.

Convention	Symbol	Description/Example
Mutually exclusive parameters	(Parentheses   and   vertical   bars)	Within a code description, bar separators denote options from which one must be chosen.  <code>% data = hdfread (start   stride   edge)</code>
Optional parameters XML variable text	[square brackets]	Within a code description, square brackets denote completely optional commands or parameters.  <code>% sed [-n, -quiet]</code>  Use square brackets in XML examples to differentiate them from tags.  <code>&lt;CustomerId&gt;[ID]&lt;/CustomerId&gt;</code>
Variables	<arrow brackets>	Within a code sample, arrow brackets denote a variable that must be replaced with a valid value.  <code>% ec2-register &lt;your-s3-bucket&gt;/image.manifest</code>