

**Flexible Payments Service  
Account Management Quick Start  
API Version 2010-08-28**

## **Flexible Payments Service Account Management Quick Start**

Amazon Web Services

Copyright © 2012 - 2013 Amazon Web Services LLC or its affiliates. All rights reserved.

The following are trademarks or registered trademarks of Amazon: Amazon, Amazon.com, Amazon.com Design, Amazon DevPay, Amazon EC2, Amazon Web Services Design, AWS, CloudFront, EC2, Elastic Compute Cloud, Kindle, and Mechanical Turk. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon.

All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

# Table of Contents

Welcome.....	1
How Do I...?.....	1
Introduction to Amazon FPS Account Management Quick Start.....	2
Overview of Amazon FPS Account Management Quick Start.....	2
Business Model.....	2
Features .....	2
Amazon Flexible Payments Service .....	3
Sender, Recipient, and Caller Actions .....	5
Request Security .....	5
Sandbox.....	6
Errors .....	6
REST Errors .....	7
Response Codes .....	7
CE and SE Status Codes.....	7
CE (Caller Exception).....	8
SE (System Error) .....	8
Business Considerations .....	8
Amazon Payments and Your Website.....	8
Supported Payment Instruments and Currencies.....	9
Amazon Payments Account.....	9
Amazon Recipient Fees.....	9
Fraud .....	9
Disputes.....	10
Amazon A-z Guarantee.....	10
Amazon Buyer Dispute Program.....	11
Chargebacks.....	12
WSDLs and Schemas .....	12
WSDL .....	13
Schema .....	13
Programming Guide.....	14

## Flexible Payments Service Account Management Quick Start

Amazon FPS Endpoints .....	14
Important Values to Store in Your Database.....	15
Caller Reference .....	15
Transaction ID .....	15
Request ID.....	15
Accepting Payments from Mobile Devices .....	15
Amazon FPS Account User Interface .....	16
Getting Your Balance .....	16
Viewing Your Account Activity.....	16
Getting Account Information Programmatically .....	17
Working with Signatures .....	17
Generating a Signature .....	18
About Signature Version 2 .....	19
Verifying the ReturnURL and IPN Notifications .....	19
Access Key Rotation.....	20
Testing Your Applications for Free.....	20
Sandbox Endpoints.....	21
Sandbox Use .....	21
Simulating a Mobile Client.....	21
Error Simulation .....	22
Testing Signatures .....	22
Migrating your Application from the Sandbox to Production .....	23
Setting Up Instant Payment Notification.....	24
Setting Up IPN Preferences .....	25
Receiving IPN Notifications.....	25
Setup Process for a Script to Receive IPN .....	25
How To Verify the IPN Signature .....	26
Common IPN Response Elements.....	26
IPN Responses for Marketplace Transactions .....	28
Email Notification Templates .....	31
Amazon FPS API Reference.....	32
Actions .....	32
GetAccountActivity.....	32

## Flexible Payments Service Account Management Quick Start

Description .....	32
Request Parameters.....	32
Response Elements .....	33
Errors .....	34
Sample REST Request .....	34
Sample Response to REST Request.....	34
Related Actions .....	40
GetAccountBalance .....	40
Description .....	40
Request Parameters.....	40
Response Elements .....	41
Errors .....	41
Sample REST Request .....	41
Sample Response to REST Request.....	41
Related Actions .....	42
GetTokens .....	42
Description .....	42
Request Parameters.....	42
Response Elements .....	42
Errors .....	43
Sample REST Request .....	43
Sample Response to REST Request.....	43
Related Actions .....	44
GetTokenUsage.....	44
Description .....	44
Request Parameters.....	44
Response Elements .....	44
Errors .....	44
Sample REST Request .....	45
Sample Response to REST Request.....	45
Related Actions .....	45
GetTransaction .....	46
Description .....	46

## Flexible Payments Service Account Management Quick Start

Request Parameters.....	46
Response Elements .....	46
Errors .....	46
Sample REST Request .....	46
Sample Response to REST Request.....	47
Related Actions .....	48
GetTransactionStatus .....	48
Description .....	48
Request Parameters.....	48
Response Elements .....	48
Status Codes.....	49
Errors .....	49
Sample REST Request .....	49
Sample Query Request .....	50
Sample Response to REST Request.....	50
VerifySignature .....	51
Description .....	51
Request Parameters.....	51
Response Elements .....	52
Errors .....	52
Sample REST Request .....	52
Sample Query Request .....	53
Sample Response to REST Request.....	53
Common Request Parameters .....	53
Common Response Elements .....	54
Errors .....	55
Data Types.....	63
Enumerated Data Types .....	63
AccountBalance .....	63
ChargeFeeTo .....	63
CurrencyCode .....	63
FPSOperation.....	64
InstrumentId .....	64

## Flexible Payments Service Account Management Quick Start

InstrumentStatus .....	64
PaymentMethod .....	64
RelationType .....	65
SortOrderByDate .....	65
TokenStatus .....	65
TokenType .....	65
TransactionalRole .....	65
TransactionStatus .....	65
TransactionStatus (IPN) .....	66
Complex Data Types .....	67
Amount .....	67
AvailableBalances .....	67
DebtBalance .....	67
DescriptorPolicy .....	67
MarketplaceRefundPolicy .....	68
RelatedTransaction .....	68
StatusHistory .....	68
Token .....	68
TokenUsageLimit .....	69
Transaction .....	69
TransactionDetail .....	70
TransactionPart .....	72
Code Samples .....	73
Understanding the Amazon FPS Samples .....	73
Understanding the VerifySignature Sample .....	74
Using the VerifySignature Sample .....	74
Locations of the VerifySignatureSample Files in Other Libraries .....	76
C# File Locations for the Amazon.FPS VerifySignature Sample .....	76
Perl File Locations for the Amazon.FPS VerifySignature Sample .....	77
PHP File Locations for the Amazon.FPS VerifySignature Sample .....	77
Understanding the Amazon CBUI Samples .....	78
Java .....	79
Co-Branded service request with Java SDK Sample .....	80

## Flexible Payments Service Account Management Quick Start

Locations of the CBUI SingleUsePipeline Files in Other Libraries .....	81
C# File Locations for the Amazon.FPS CBUI Sample.....	81
Perl File Locations for the Amazon.FPS VerifySignature Sample .....	82
PHP File Locations for the Amazon.FPS VerifySignature Sample .....	82
Understanding the IPNAndReturnURLValidation Sample .....	82
Locations of the IPNAndReturnURLValidation Files in Other SDKs .....	84
C# File Locations for the Amazon.IpnReturnUrlValidationSample Library.....	84
Perl File Locations for the IpnReturnUrlValidation Library.....	84
PHP File Locations for the IpnReturnUrlValidation Library.....	84
Getting the Samples .....	85
Amazon FPS Resources .....	86
Glossary.....	87
Document History .....	90

## Welcome

# Welcome

This is the Amazon FPS Account Management Quick Start. This guide describes the concepts for and gives instructions to set up programmatic access to buyer and developer account activity.

Amazon Flexible Payments Service is a web service that enables developers to accept payments on their website. The payments can be for selling goods or services, raise donations, execute recurring payments, and send payments.

## How Do I...?

How do I...?	Topics
Decide whether Amazon FPS is right for my needs:	<a href="#">Amazon FPS</a> detail page
Get started with Amazon FPS API quickly	<a href="#">Amazon Flexible Payments Service Getting Started Guide</a>
Learn about the Amazon Flexible Payments Service features, and key concepts related to Account Management Quick Start	Introduction to Amazon FPS Account Management Quick Start
Learn about the Amazon FPS user interface, programmatic retrieval of account information, working with signatures, testing, and sample applications	Amazon FPS Account User Interface
Obtain account information programmatically using the Amazon FPS API	Getting Account Information Programmatically
Find reference information on the actions provided by this quick start	Amazon FPS API Reference
Find examples for creating signatures and making a pay request	Code Samples

# Introduction to Amazon FPS Account Management Quick Start

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

## Overview of Amazon FPS Account Management Quick Start

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

### Business Model

The Amazon FPS Account Management Quick Start provides detailed information about accounts, transactions, and payment tokens. You can use this functionality to keep track of your transactions and account activities or to create a console that provides similar information either for your buyers or the sellers.

### Features

Amazon FPS Account Management Quick Start provides the following major features:

**Get account activity** —You can retrieve detailed information about one account for a specified time period and customize the results using various request parameters.

**Get detailed transaction information**—You can get detailed information about a specified transaction.

**Get token information**—You can get detailed information about some or all of the payment tokens you use.

For different functionality, such as multi-use payment tokens, go to one of the other Amazon FPS Quick Starts.

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

## Introduction to Amazon FPS Account Management Quick Start

**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, or services.

**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.

### Note

In this guide, "you" refers to the caller developing the website or application.

## Amazon Flexible Payments Service

The Quick Start implementation covered in this guide is one of four 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 website 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 website.

## Introduction to Amazon FPS Account Management Quick Start

**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 website 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 four Quick Start implementations, each providing a different slice of Amazon FPS functionality:

How do I...?	Topics
<a href="#">Basic</a>	Facilitates one-time payment between a buyer and a developer who is also the merchant for e-commerce, digital content, donations, services.
<a href="#">Marketplace</a>	Facilitates one-time payment between buyer and merchant where you are a third party developer, a caller, who hosts the merchant's products and order pipeline. 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.
<a href="#">Advanced</a>	Facilitates multiple or recurring payments.
<a href="#">Account Management</a>	Accesses buyer and developer account activity programmatically. Alternatively, view account activity and balances can be viewed on the <a href="#">Amazon Payments website</a> .

## Introduction to Amazon FPS Account Management Quick Start

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

# 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=
```

## Introduction to Amazon FPS Account Management Quick Start

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 (signature version 1) was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. Whenever you sign a request with your access keys, you must now use signature version 2.

Signing is required for all Amazon FPS API requests, 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 has been tampered. For detailed information about generating a signature, see “Working with Signatures.”

## 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 website.

The sandbox separate endpoints for the Amazon FPS API 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 Signing Up for the Amazon FPS Sandbox in the [Amazon Flexible Payments Service Getting Started Guide](#).

## Errors

Amazon FPS error results provide information about syntax 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.

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 “Errors.”

# 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.”

## 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 nontransient 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

## Introduction to Amazon FPS Account Management Quick Start

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 displays an error message on the user interface describing the problem. If you click the provided **Continue** button, the CBUI returns you (as the test buyer) to your website (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.
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 temporally failed in Amazon FPS. You can retry the request again.

## Business Considerations

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

## Amazon Payments and Your Website

You can add an Amazon Payments icon to your website to let your buyers know you accept Amazon Payments. For more information, see the [Payment Marks and Graphics](#) page on the Amazon Payments website.

# Supported Payment Instruments and Currencies

Amazon FPS supports the following payment instruments:

- Amazon Payments account balance
- Bank account debits (ABT)
- 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 websites. 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 website 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 account 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 website.

## 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 website. 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

## Introduction to Amazon FPS Account Management Quick Start

requirements of the policy. You could still be held liable for service chargebacks. Please see our [User Agreement](#) for details.

## 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

## Introduction to Amazon FPS Account Management Quick Start

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 they submit their 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 nonphysical 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, see [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, see [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 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 website. From the transaction or order details page, they can also click **Problem with this transaction?** or **Problem with this order** to file 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, Amazon 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 turn 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 will work with you and the buyer to resolve the chargeback. You will 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 website, 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 an order will ship and keep them informed about the progress of their order.
  - 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. To enable computers running different operating systems to 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.

## Introduction to Amazon FPS Account Management Quick Start

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/2010-08-28/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/2010-08-28/AmazonFPS.xsd>.

# Programming Guide

This programming guide provides task-oriented descriptions of how to use and implement Amazon FPS Account Management Quick Start actions. For a complete description of those actions, see the [Amazon FPS API Reference](#).

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](#).

How do I?	Relevant Sections
Get account information non-programmatically using the Amazon FPS Account User Interface	Amazon FPS Account User Interface
Retrieve account information programmatically using the Amazon FPS API	Getting Account Information Programmatically
Create request signatures	Working with Signatures
Test your application using the Amazon FPS sandbox	Testing Your Applications for Free
Use the Amazon FPS Samples	Code Samples

## Amazon FPS Endpoints

Amazon FPS has four endpoints where you send requests, listed in the following table. Two are for sandbox testing of CBUI and API requests, and two are for production Co-Branded User Interface (CBUI) and API requests.

Endpoint	Purpose
<a href="https://authorize.payments-sandbox.amazon.com/cobranded-ui/actions/start">https://authorize.payments-sandbox.amazon.com/cobranded-ui/actions/start</a>	Sandbox endpoint for Co-Branded service requests.
<a href="https://authorize.payments.amazon.com/cobranded-ui/actions/start">https://authorize.payments.amazon.com/cobranded-ui/actions/start</a>	Production endpoint for Co-Branded service requests.
<a href="https://fps.sandbox.amazonaws.com">https://fps.sandbox.amazonaws.com</a>	Sandbox endpoint for Amazon FPS actions
<a href="https://fps.amazonaws.com">https://fps.amazonaws.com</a>	Production endpoint for Amazon FPS actions.

# Important Values to Store in Your Database

When you use Amazon FPS, there are times when you should store important information in your database. This topic describes 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. Amazon FPS uses the caller reference value to provide request idempotency for a seven-day period.

### 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 "RepeatedAPIRequests" in the [Amazon Flexible Payments Service Advanced Quick Start](#)).

## Request ID

Amazon FPS returns a RequestId for each Amazon FPS API call accepted for processing. If you have a problem with a request, AWS will ask for the request ID to troubleshoot the issue. For more information about RequestId, see "Common Response Elements" in the [Amazon FPS API Reference](#).

# Accepting Payments from Mobile Devices

Amazon FPS provides a seamless integration with web sites optimized for mobile devices. No special Amazon FPS coding is required. The software detects the client browser HTTP\_USER\_AGENT and routes the request along the appropriate CBUI pipeline. A separate pipeline is optimized for the mobile device experience.

## Programming Guide

The Amazon Payments service has been designed and developed for use within a web browser only. Our service cannot be used within a native application (including, without limitation, iOS, Android, RIM and Windows operating systems). Amazon Payments reserves the right to suspend the Payment Account of any user of our services that has implemented our Services within a native application.

### Note

For optimal security, your mobile application should make use of a full browser instance (not a browser embedded within an application). It should display the browser address bar to enable customers to confirm the URL.

The user CBUI experience is managed for you. You handle all Amazon FPS requests, return URLs, and IPN notifications regardless of which client browser the customer is using.

Amazon FPS makes it easy to test your mobile client experience. For more information, see “Simulating a Mobile Client.”

## Amazon FPS Account User Interface

Amazon FPS provides a user interface at <http://payments.amazon.com> where you can view your account and transaction history. This section covers that interface. For information about programmatic access to the account and transaction history, see “Getting Account Information Programmatically.”

## Getting Your Balance

Your account balance shows you the amount of funds you have in your Amazon Payments Account. There are two types of balances: the account balance and the disbursable balance.

The account balance is the total amount of money in your account, including debit transactions that are not yet completed. The disbursable balance is the total amount of money available for immediate transactions and withdrawals.

### To view your account balance

1. Go to <http://payments.amazon.com>.
2. Click the **Your Account** tab.

The **Your Account** page appears. On it is your account balance.

## Viewing Your Account Activity

Your account activity includes information for each transaction, such as the amount paid, when it was paid, to whom, the status of the payment, any fees associated with the transaction, and so forth.

## Programming Guide

To view your account activity

1. Go to <http://payments.amazon.com>.
2. Click the **Your Account** tab.

The **Account Activity** page appears.

From here you can do a simple search based on the **Activity:** and **Within:** drop-down menus.

You can also click **Advanced Search Options** and display a page that lets you specify a particular date range.

3. Make your selection, click **View**, and the results will appear.

## Getting Account Information Programmatically

Amazon FPS enables you to retrieve account transaction history programmatically. You can reconstruct the transaction history to assist inquiring senders or recipients who have used you as a caller.

The following table describes the tasks you can perform programmatically and which API action to use.

To do this...	Use this action
Get your current account balance.	GetAccountBalance
Get account activity, filtered by time period, action used, payment type, transaction status, etc.	GetAccountActivity
Get details about a specific transaction.	GetTransaction
Get the current status of a particular transaction.	GetTransactionStatus
Get all or a subset of your tokens.	GetTokens
Get a list of the limitations that you have set on a specific multi-use token.	GetTokenUsage
Check the signature of a return URL response or IPN notification.	VerifySignature

## Working with Signatures

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.”

# 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).
  - 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).

#### 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.

```
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 (/).

## Programming Guide

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 the Amazon FPS action, VerifySignature, which enables you to securely check a response using a server-side call.

### Important

The original implementation of signature version 2 supported client-side signature validation using PKI. Client-side signature validation was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. If you have been using client-side signature validation, you must switch to server-side validation using the FPS action VerifySignature.

Signature version 2 supports AWS access key rotation, further enhancing the security of your button content. For more information, see "Access Key Rotation."

### Important

The previous method for signing (signature version 1) was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. Whenever you sign a request with your access keys, you must now use signature version 2.

## 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.

## Programming Guide

Signature version 2 security enables you to verify the signature of the response using a server-side call to the VerifySignature FPS Action. To use it, modify your returnUrl and ipnUrl pages to parse the notification. From those components, you assemble the relevant parameters for VerifySignature and sign it like any other request. The result of the call is either Success, meaning the response is valid, or Failure, indicating the response is suspect.

For more information on VerifySignature, see “VerifySignature.” In addition, you can use the validation samples to assist creating your own validation pages. For more information, see “Understanding the IPNAndReturnURLValidation Sample.”

### Important

The original implementation of signature version 2 supported client-side signature validation using PKI. Client-side signature validation was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. If you have been using client-side signature validation, you must switch to server-side validation using the FPS action VerifySignature.

## 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 website at <http://aws.amazon.com>) enables you to create a second set, and allows you to activate and deactivate the sets independently.

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.

## Testing Your Applications for Free

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

## Programming Guide

- 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>. You must be logged in to view this page.

## 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>
- Central FPS Sandbox Resource page— <http://docs.amazonwebservices.com/AmazonFPS/latest/SandboxLanding/index.html>

## 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
- Tracking the cumulative effect of a series of Pay calls. While you can't adjust the time/date of the call, you can check that the values change as expected in your test account(s) with each transaction.

## Simulating a Mobile Client

You can easily test the CBUI pipeline that your customers experience when they use their mobile devices. Amazon FPS uses the value of the client browser's HTTP\_USER\_AGENT to route the request along the appropriate pipeline. If you set your development environment to report a value for HTTP\_USER\_AGENT reported by a mobile device, Amazon FPS will invoke the mobile pipeline.

For example, the following value simulates an Apple iPhone 3G version 2.1, with Safari 3.1.1:

## Programming Guide

```
Mozilla/5.0 (iPhone; U; CPU iPhone OS 2_1 like Mac OS X; en-us)
AppleWebKit/525.18.1 (KHTML, like Gecko)
Version/3.1.1.1 Mobile/5F136 Safari/525.20
```

## 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	Z1LGRXR4HMDZBSFKXELA32KZASGWD8IHMHZ CK4DETR784LDLD1GMFW4P3WT8VTGX
Email address not verified	E3FR7BARJV3PB631PMKV74PGKCJLBHI1Q1K MQN7BJ2JJICPDKN3N1CJIKFZ8D7NN
Suspended account	H216UECZ8ZM1G8G4QA3V7RKF8JDFZ9SI3SJA FSGUKBBNDHX1NVM8GUQRZNRNAHER

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](#). Each example contains a signature calculated from the values in the rest of the example.

## Programming Guide

1. Copy any one of the sample query request examples from among the Actions in [Amazon FPS API Reference](#).
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.
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:

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 <https://payments.amazon.com/sdui/sdui/managecobranding>.
3. Please ensure you have specified your co-branding URL on production using the form at <https://payments.amazon.com/sdui/sdui/managecobranding>.
4. Please ensure that the rest of your account settings are current at <https://payments.amazon.com/sdui/sdui/accountsettings>.

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 email address and password.

# 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.

Since you cannot easily determine when asynchronous transactions 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

**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 website 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 appears.
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), 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 Common IPN Response Elements).
3. Write your program so that it verifies the signature value sent in the IPN to make sure Amazon FPS sent the IPN. For more information, see “Verifying the ReturnURL and IPN Notifications”, below.
4. Write your program to use the returned elements to notify you of the IPN-related transactions.

# 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 server-side signature verification. For more information, see “Verifying the ReturnURL and IPN Notifications.”

You can use the IPNAndReturnURLValidation sample to assist creating your own IPN validation page. For more information, see “Understanding the IPNAndReturnURLValidation Sample.”

## Common IPN Response Elements

These IPN response elements are common to most types of transactions. For a list of IPN response elements for marketplace transactions, see “IPN Responses for Marketplace Transactions.”

Name	Description
addressFullName	Full name of the buyer/sender. Type: String
addressLine1	Sender's address (first line). For IPN, this element is returned only if the value has been updated with Amazon. Type: String
addressLine2	Sender's address (second line). For IPN, this element is returned only if the value has been updated with Amazon. Type: String
addressState	Sender's state. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
addressZip	Sender's post code. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
addressCountry	Sender's country. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
addressPhone	Sender's phone number. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
buyerEmail	Sender's email address.  <b>Note:</b> The buyerEmail element is not returned when the recipient is not the caller (i.e., marketplace transactions).

## Programming Guide

Name	Description
	Type: String Size: 65 bytes
buyerName	Sender's name. Type: String Size: 128 bytes
certificateUrl	A url specifying the location of the certificate used for signing the response. Type: String Max Size: 1024 bytes
customerEmail	Customer's email address. Type: String Size: 65 bytes
customerName	Buyer/Sender Full Name. Type: String Size: 128 bytes
dateInstalled	If the notificationType element (below) is TokenCancellation, this element contains the date the token was installed. Type: String Size: 30 bytes
isShippingAddressProvided	If the IPN results include address updates, this element contains TRUE. Otherwise this element is not present in the response. Type: String
operation	The name of the payment action, also called an operation, used for this transaction. Type: String Max Size: 20 bytes
notificationType	Notification type may be either TokenCancellation or TransactionStatus Type: String Size: 20 bytes
paymentMethod	The payment method used by the sender. For more information, see the "IPN values in PaymentMethod." Type: String Size: 20 bytes
paymentReason	Reason for payment. Type: String
recipientEmail	Recipient's email address. <b>Note</b> As a security precaution, you should always check that the recipient email is the same as the one in your original request. Type: String Size: 65 bytes
recipientName	Recipient's name.

## Programming Guide

Name	Description
	Type: String Size: 128 bytes
signature	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 “Handling the Receipt of IPN Notifications.” We recommend that you always verify the signature using the method in How to Verify the IPN Signature. Type: String Size: 512 bytes
signatureVersion	A value that specifies the Signature format. Type: Integer Valid Values: 2
signatureMethod	A value that specifies the signing method. Type: String Valid Values: HmacSHA256 (preferred) and HmacSHA1.
tokenId	If notificationType is TokenCancellation, this element contains the ID of the cancelled token. Type: String Size: 65 bytes
tokenType	If notificationType is TokenCancellation, this element contains the type of the canceled token. Type: String Size: 20 bytes
transactionAmount	Specifies the amount payable in this transaction; for example, USD 10.00. Type: String Size: 30 bytes
transactionDate	The date when this transaction occurred, specified in seconds since the start of the epoch. Type: Long Size: 40 bytes
transactionId	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. Type: String Size: 35 bytes
transactionStatus	Specifies the status of the transaction. For more information, see “TransactionStatus (IPN).” Type: String

## IPN Responses for Marketplace Transactions

The following IPN response elements are returned only for marketplace transactions.

## Programming Guide

### IPN Marketplace Transaction Elements

Name	Description
buyerName	Sender's name. Type: String
operation	The name of the payment action, also called an operation, used for this transaction. Type: String Max Size: 20 bytes
paymentMethod	The payment method used by the sender. For more information, see the "IPN values in PaymentMethod." Type: String
paymentReason	Reason for payment. Type: String
recipientEmail	Recipient's email address. Type: String
recipientName	Recipient's name. Type: String
referenceId	If you specified a referenceId in the button creation form, Amazon Payments returns the referenceId to you. Type: String
signature	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 "Handling the Receipt of IPN Notifications." We recommend that you always verify the signature using the method in How to Verify the IPN Signature. Type: String
status	Specifies the status of the transaction. For more information, see "TransactionStatus (IPN)." Type: String
transactionAmount	Specifies the amount payable in this transaction; for example, USD 10.00. This element is not being returned in the current version. Type: Double
transactionDate	The date when this transaction occurred, specified in seconds since the beginning of the epoch. Type: Long
transactionId	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. Type: String

## Programming Guide

## Programming Guide

# Email Notification Templates

Many transactions generate email messages from Amazon Payments, sent to either the buyer, seller, or web site owner in the case of a marketplace transaction. Transaction details are listed in the body of the email message. The content of the email message sent out depends on the transaction and its status.

The table here defines the templates that are used, and provides a link to an example message for each.

Email Template Name	Description
<a href="#">BAVerifFailed</a>	Bank account verification failed
<a href="#">BAVerifStartedForPersonalBusiness</a>	Bank account verification started for personal and business account
<a href="#">BAVerifStartedForDeveloper</a>	Bank account verification started for developer account
<a href="#">BAVerifSuccess</a>	Bank account verification successful
<a href="#">CCVerifFailedBusiness</a>	Credit card verification failed for business account
<a href="#">ConfirmEmailPersonal</a>	Email confirmation sent to confirm the email address for the personal account.
<a href="#">ConfirmEmailDeveloper</a>	Email confirmation sent to confirm the email address for a developer account.
<a href="#">ConfirmEmailBusiness</a>	Email confirmation sent to confirm the email address for a business account.
<a href="#">DailySummary</a>	Daily summary of transactions
<a href="#">DepositFailure</a>	Deposit failed
<a href="#">DepositFundsInitiated</a>	Request to deposit funds initiated
<a href="#">DepositSuccess</a>	Deposit successful
<a href="#">DeveloperRegistrationCompleted</a>	Developer registration complete
<a href="#">MonthlyNotif</a>	Notification for monthly statement
<a href="#">UpgradePersonalToBusiness</a>	Upgrade to business account successful
<a href="#">VerifyEmailSuccessPersonal</a>	Email address verified for personal account
<a href="#">VerifyEmailSuccessBusiness</a>	Email address verified for business account
<a href="#">WithdrawFailure</a>	Withdraw failed, bank unable to accept electronic transaction
<a href="#">WithdrawFundsInitiated</a>	Withdraw funds has been initiated

# Amazon FPS API Reference

The following sections of the guide provide API reference material for the Amazon FPS API.

The current version of the Amazon FPS API is 2010-08-28.

The WSDL is located at <https://fps.amazonaws.com/doc/2010-08-28/AmazonFPS.wsdl>.

The schema is located at <https://fps.amazonaws.com/doc/2010-08-28/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](#).

## Actions

This section describes the actions that are available with Amazon FPS Account Management Quick Start.

### GetAccountActivity

**Description**

The GetAccountActivity action returns transactions from an account for a given time period. You can further customize the results using the other request parameters.

**Request Parameters**

Parameter	Description	Required
EndDate	Specifies the final date for the list of transactions to return. If endDate is not specified, Amazon FPS returns transactions to the current date. Type: dateTime Default: Current date	No
FPSOperation	Filters the results by Amazon FPS action. For example, the value Pay returns only transactions involving the Pay action. Type: FPSOperation Default: None	No
MaxBatchSize	Specifies the maximum number of transactions returned in the response.	No

## Amazon FPS API Reference

Parameter	Description	Required
	Type: Integer Default: 200 Constraints: Between 20 and 200	
PaymentMethod	Specifies the payment method, such as CC or ABT. Type: PaymentMethod Default: None	No
ResponseGroup	Subheading that allows you to group the responses. Type: String Default: None	No
Role	List of roles arranged in sort order based on the role: sender, recipient, or caller. Type: TransactionalRole Default: None	No
SortOrderByDate	Specifies how to sort the results and in what order. The date used is the date the request was received by Amazon FPS. Type: SortOrderByDate Default: Descending Valid Values: Descending   Ascending	No
StartDate	Specifies the first date of transactions to return. Type: dateTime Constraints: Present, past dates	Yes
Status	Filters the results based on the transaction status. Type: TransactionStatus Default: None	No

You must also use parameters that are common to all requests that are described in Common Request Parameters. The common parameters must be explicitly added in REST calls. Parameter names are case sensitive.

### Response Elements

Element	Description
BatchSize	Specifies the total number of results returned. This element is always returned. Type: Integer
StartTimeForNextTransaction	Provides the start time for the next transaction. Amazon FPS returns a maximum of 200 results for one request. You can use the value returned by this parameter as the start time/date for the next request.  For example, if you need the account activity for the period Jan-1-2010 to Dec-31-2010 and there are more than 200

## Amazon FPS API Reference

Element	Description
	transactions during that period, Amazon FPS returns a maximum of 200 transactions and sends the date, Apr-20-2010, which is the date of the next transaction to be returned. You can get the next 200 transactions using Apr-20-2010 as the new start date. This element is returned only if there are more than 200 transactions. Type: dateTime
Transaction	Specifies the list of transactions. This element is returned only if there are transactions. Type: Transaction

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

### Errors

This action can return the following errors:

- AccessFailure
- AccountClosed
- AuthFailure
- InternalError
- InvalidClientTokenId
- InvalidDateRange
- InvalidParams
- SignatureDoesNotMatch

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=GetAccountActivity
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Signature=Bs3etBhuZ2Huf8gxLO0EaG4evxq%2BjzT2Bjg6YMAF3RE%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2&Version=2008-09-17
&StartDate=2009-10-07Z
&Timestamp=2009-10-07T11%3A14%3A56.406Z
&Version=2008-09-17
```

### Sample Response to REST Request

```
<GetAccountActivityResponse
  xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <GetAccountActivityResult>
    <BatchSize>5</BatchSize>
    <Transaction>
      <TransactionId>
        14GN2BUHUAV4KG5S8USHN79PQH1NGN5ADK4
```

## Amazon FPS API Reference

```
</TransactionId>
<CallerTransactionDate>
  2009-10-07T01:37:54.765-07:00
</CallerTransactionDate>
<DateReceived>2009-10-07T01:38:11.262-07:00</DateReceived>
<DateCompleted>2009-10-07T01:38:12.857-07:00</DateCompleted>
<TransactionAmount>
  <CurrencyCode>USD</CurrencyCode>
  <Value>1.000000</Value>
</TransactionAmount>
<FPSOperation>Pay</FPSOperation>
<TransactionStatus>Success</TransactionStatus>
<StatusMessage>
  The transaction was successful and the payment instrument
  was charged.
</StatusMessage>
<StatusCode>Success</StatusCode>
<TransactionPart>
  <Role>Recipient</Role>
  <Name>Test Business</Name>
  <FeesPaid>
    <CurrencyCode>USD</CurrencyCode>
    <Value>0.100000</Value>
  </FeesPaid>
</TransactionPart>
<TransactionPart>
  <Role>Caller</Role>
  <Name>Test Business</Name>
  <Reference>CallerReference10</Reference>
  <Description>MyWish</Description>
  <FeesPaid>
    <CurrencyCode>USD</CurrencyCode>
    <Value>0.000000</Value>
  </FeesPaid>
</TransactionPart>
<PaymentMethod>CC</PaymentMethod>
<SenderName>Test Business</SenderName>
<CallerName>Test Business</CallerName>
<RecipientName>Test Business</RecipientName>
<FPSFees>
  <CurrencyCode>USD</CurrencyCode>
  <Value>0.100000</Value>
</FPSFees>
<Balance>
  <CurrencyCode>USD</CurrencyCode>
  <Value>7.400000</Value>
</Balance>
<SenderTokenId>
563INMLCG3ZJJ4L1I7BB31MN2FBQUCVXNTDRTCT5A2DJDYG6LNZ7KSNUJPI7TVIF
</SenderTokenId>
</Transaction>
```

## Amazon FPS API Reference

```
<Transaction>
  <TransactionId>
    14GN105992IEOB3ELM1SCUFTSOQ3C6S7NR2
  </TransactionId>
  <CallerTransactionDate>
    2009-10-07T01:27:21.469-07:00
  </CallerTransactionDate>
  <DateReceived>2009-10-07T01:27:22.793-07:00</DateReceived>
  <DateCompleted>2009-10-07T01:27:23.335-07:00</DateCompleted>
  <TransactionAmount>
    <CurrencyCode>USD</CurrencyCode>
    <Value>4.000000</Value>
  </TransactionAmount>
  <FPSOperation>Pay</FPSOperation>
  <TransactionStatus>Success</TransactionStatus>
  <StatusMessage>
    The transaction was successful and the payment instrument
    was charged.
  </StatusMessage>
  <StatusCode>Success</StatusCode>
  <TransactionPart>
    <Role>Recipient</Role>
    <Name>Test Business</Name>
    <Reference>CC Digital Download - 1254904041469</Reference>
    <Description>CC Digital Download</Description>
    <FeesPaid>
      <CurrencyCode>USD</CurrencyCode>
      <Value>0.000000</Value>
    </FeesPaid>
  </TransactionPart>
  <TransactionPart>
    <Role>Caller</Role>
    <Name>Test Business</Name>
    <Reference>CC Digital Download - 1254904034205</Reference>
    <Description>
      CC Digital Download - payment for mp3 from digital.
    </Description>
    <FeesPaid>
      <CurrencyCode>USD</CurrencyCode>
      <Value>0.000000</Value>
    </FeesPaid>
  </TransactionPart>
  <PaymentMethod>CC</PaymentMethod>
  <SenderName>Test Business</SenderName>
  <CallerName>Test Business</CallerName>
  <RecipientName>Test Business</RecipientName>
  <FPSFees>
    <CurrencyCode>USD</CurrencyCode>
    <Value>0</Value>
  </FPSFees>
  <Balance>
```

## Amazon FPS API Reference

```
<CurrencyCode>USD</CurrencyCode>
<Value>6.500000</Value>
</Balance>
<SenderTokenId>
513I1MGCG6ZZJ49157BZ3EMNJFAQU6V9NTSRUCTEANDJ3X46LGZKNKSJUVPIXTPID
</SenderTokenId>
<RecipientTokenId>
D639FT4TMP4QK9UBH6PAK2WAXGHDZSBUX3UJSGVX3LEFVVGU7XDQXMENL4OGVZEGB
</RecipientTokenId>
</Transaction>
<Transaction>

<TransactionId>
14GN1NHHN489BFGH6D8BMGT8NLSR2DJ4PNK
</TransactionId>
<CallerTransactionDate>
2009-10-07T01:26:58.190-07:00
</CallerTransactionDate>
<DateReceived>2009-10-07T01:27:02.583-07:00</DateReceived>
<DateCompleted>2009-10-07T01:27:04.435-07:00</DateCompleted>
<TransactionAmount>
<CurrencyCode>USD</CurrencyCode>
<Value>5.000000</Value>
</TransactionAmount>
<FPSOperation>Pay</FPSOperation>
<TransactionStatus>Success</TransactionStatus>
<StatusMessage>
The transaction was successful and the payment instrument
was charged.
</StatusMessage>
<StatusCode>Success</StatusCode>
<TransactionPart>
<Role>Caller</Role>
<Name>Test Business</Name>
<Reference>CC Digital Download - 1254903995419</Reference>
<FeesPaid>
<CurrencyCode>USD</CurrencyCode>
<Value>0.000000</Value>
</FeesPaid>
</TransactionPart>
<TransactionPart>
<Role>Recipient</Role>
<Name>Test Business</Name>
<FeesPaid>
<CurrencyCode>USD</CurrencyCode>
<Value>0.300000</Value>
</FeesPaid>
</TransactionPart>
<PaymentMethod>CC</PaymentMethod>
<SenderName>Test Business</SenderName>
<CallerName>Test Business</CallerName>
```

## Amazon FPS API Reference

```
<RecipientName>Test Business</RecipientName>
<FPSFees>
  <CurrencyCode>USD</CurrencyCode>
  <Value>0.300000</Value>
</FPSFees>
<Balance>
  <CurrencyCode>USD</CurrencyCode>
  <Value>6.500000</Value>
</Balance>
<SenderTokenId>
513ISM2CGDZPJ4S1D7BH3HMNIFCQUAVNNTQRXCTHAUDJLXV6LMZLKSTUKPITTXIV
</SenderTokenId>
</Transaction>
<Transaction>
  <TransactionId>
    14GMNT2PDVUJA18L44TO4DIFJEJRF9LTV2T
  </TransactionId>
  <CallerTransactionDate>
    2009-10-06T22:35:02.031-07:00
  </CallerTransactionDate>
  <DateReceived>2009-10-06T22:35:18.317-07:00</DateReceived>
  <DateCompleted>2009-10-06T22:35:19.332-07:00</DateCompleted>
  <TransactionAmount>
    <CurrencyCode>USD</CurrencyCode>
    <Value>1.000000</Value>
  </TransactionAmount>
  <FPSOperation>Refund</FPSOperation>
  <TransactionStatus>Success</TransactionStatus>
  <StatusMessage>
    The transaction was successful and the payment instrument
    was charged.
  </StatusMessage>
  <StatusCode>Success</StatusCode>
  <TransactionPart>
    <Role>Caller</Role>
    <Name>Test Business</Name>
    <Reference>CallerReference09</Reference>
    <FeesPaid>
      <CurrencyCode>USD</CurrencyCode>
      <Value>0.000000</Value>
    </FeesPaid>
  </TransactionPart>
  <TransactionPart>
    <Role>Sender</Role>
    <Name>Test Business</Name>
    <FeesPaid>
      <CurrencyCode>USD</CurrencyCode>
      <Value>-0.100000</Value>
    </FeesPaid>
  </TransactionPart>
  <PaymentMethod>CC</PaymentMethod>
```

## Amazon FPS API Reference

```
<SenderName>Test Business</SenderName>
<CallerName>Test Business</CallerName>
<RecipientName>Test Business</RecipientName>
<FPSFees>
  <CurrencyCode>USD</CurrencyCode>
  <Value>-0.100000</Value>
</FPSFees>
<Balance>
  <CurrencyCode>USD</CurrencyCode>
  <Value>1.800000</Value>
</Balance>
</Transaction>
<Transaction>
  <TransactionId>
    14GMNRDSJ6TJTNDUTOUA917PIFJDsgNB2JP
  </TransactionId>
<CallerTransactionDate>
  2009-10-06T22:34:24.053-07:00
</CallerTransactionDate>
<DateReceived>2009-10-06T22:34:24.147-07:00</DateReceived>
<DateCompleted>2009-10-06T22:34:25.223-07:00</DateCompleted>
<TransactionAmount>
  <CurrencyCode>USD</CurrencyCode>
  <Value>1.000000</Value>
</TransactionAmount>
<FPSOperation>Pay</FPSOperation>
<TransactionStatus>Success</TransactionStatus>
<StatusMessage>
  The transaction was successful and the payment instrument
  was charged.
</StatusMessage>
<StatusCode>Success</StatusCode>
<TransactionPart>
  <Role>Recipient</Role>
  <Name>Test Business</Name>
  <Description>SubscriptionTesting</Description>
  <FeesPaid>
    <CurrencyCode>USD</CurrencyCode>
    <Value>0.100000</Value>
  </FeesPaid>
</TransactionPart>
<TransactionPart>
  <Role>Caller</Role>
  <Name>Test Business</Name>
  <Reference>63314e32-d6b0-4abd-a0ab-7b89717ba5cb</Reference>
  <FeesPaid>
    <CurrencyCode>USD</CurrencyCode>
    <Value>0.000000</Value>
  </FeesPaid>
</TransactionPart>
<PaymentMethod>CC</PaymentMethod>
```

## Amazon FPS API Reference

```
<SenderName>Test Business</SenderName>
<CallerName>Test Business</CallerName>
<RecipientName>Test Business</RecipientName>
<FPSFees>
  <CurrencyCode>USD</CurrencyCode>
  <Value>0.100000</Value>
</FPSFees>
<Balance>
  <CurrencyCode>USD</CurrencyCode>
  <Value>2.700000</Value>
</Balance>
<SenderTokenId>
533I1M9CGUZ9J4M197BM3LMNKFVQUFVFNT5RRCT2ACDJBXV6LRZ6KSRUSPI6T3I3
</SenderTokenId>
<RecipientTokenId>
D139MTVMTK4CK9QB26PKKLWA1GHDZGBGX3SJLGVU37EFUGJ7XVQIMETLSOGAZJGV
</RecipientTokenId>
</Transaction>
</GetAccountActivityResult>
<ResponseMetadata>
<RequestId>87e1570a-ef8c-4846-8265-74d07a6a83fb:0</RequestId>
</ResponseMetadata>
</GetAccountActivityResponse>
```

### Related Actions

- [GetTokenUsage](#)
- [GetTokens](#)

## GetAccountBalance

### Description

The `GetAccountBalance` action returns the current balance of your account.

### Request Parameters

This action requires only the parameters that are common to all requests. They are described in [Common Request Parameters](#). The common parameters must be explicitly added in REST calls. Parameter names are case sensitive.

## Amazon FPS API Reference

### Response Elements

Element	Description
AccountBalance	Specifies the current balance. Type: String

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

### Errors

This action can return the following errors:

- AccessFailure
- AuthFailure
- InternalError
- InvalidClientTokenId
- InvalidParams
- SignatureDoesNotMatch

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=GetAccountBalance
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Signature=FyQVfGnvleChBRKrWY9XpyXTDfQ09oSdlnGBKw4527Y%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-07T11%3A15%3A46.546Z
&Version=2008-09-17
```

### Sample Response to REST Request

```
<GetAccountBalanceResponse
  xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <GetAccountBalanceResult>
    <AccountBalance>
      <TotalBalance>
        <CurrencyCode>USD</CurrencyCode>
        <Value>7.400000</Value>
      </TotalBalance>
      <PendingInBalance>
        <CurrencyCode>USD</CurrencyCode>
        <Value>0.000000</Value>
      </PendingInBalance>
      <PendingOutBalance>
        <CurrencyCode>USD</CurrencyCode>
        <Value>0.000000</Value>
      </PendingOutBalance>
      <AvailableBalances>
```

## Amazon FPS API Reference

```
<DisburseBalance>
  <CurrencyCode>USD</CurrencyCode>
  <Value>7.400000</Value>
</DisburseBalance>
<RefundBalance>
  <CurrencyCode>USD</CurrencyCode>
  <Value>7.400000</Value>
</RefundBalance>
</AvailableBalances>
</AccountBalance>
</GetAccountBalanceResult>
<ResponseMetadata>
  <RequestId>7b74a504-7517-4d81-8312-1427570d028c:0</RequestId>
</ResponseMetadata>
</GetAccountBalanceResponse>
```

### Related Actions

- [GetTransaction](#)

## GetTokens

### Description

The GetTokens action returns all or a subset of the tokens that you installed on your account.

### Request Parameters

Parameter	Description	Required
CallerReference	A value you provide that uniquely identifies the request. For more information, see “Important Values to Store in Your Database.” Type: String Default: None Constraint: Max size = 128 characters	No
TokenStatus	Filters the results based on the status of the token. Type: TokenStatus Default: None	No
TokenType	Filters the result based on the token type. Type: TokenType Default: None	No

You must also use parameters that are common to all requests that are described in Common Request Parameters. The common parameters must be explicitly added in REST calls. Parameter names are case sensitive.

### Response Elements

Element	Description
Token	The list of the caller's tokens.

## Amazon FPS API Reference

Type: Token

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

### Errors

This action can return the following errors:

- AccessFailure
- AccountClosed
- AuthFailure
- InvalidClientTokenId
- InternalError
- InvalidParams
- SignatureDoesNotMatch

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=GetTokens
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&CallerReference=CallerReference12
&Signature=Dzp4usKpQujx9x74Wfx15BO2C3ID65P1Eb2MXwkyV8M%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-07T11%3A38%3A11.796Z
&Version=2008-09-17
```

### Sample Response to REST Request

```
<GetTokensResponse xmlns="http://fps.amazonaws.com/doc/2008-09-
17/">
  <GetTokensResult>
    <Token>
      <TokenId>
D439DTSTMP4FK9NBL6PEKZWAPGRDZ2BDX3MJNGVX37EF3GA7XRQHMEELQOGFZ9GK
      </TokenId>
      <TokenStatus>Active</TokenStatus>
      <DateInstalled>2009-10-07T04:37:57.375-07:00</DateInstalled>
      <CallerReference>CallerReference12</CallerReference>
      <TokenType>SingleUse</TokenType>
      <OldTokenId>
D439DTSTMP4FK9NBL6PEKZWAPGRDZ2BDX3MJNGVX37EF3GA7XRQHMEELQOGFZ9GK
      </OldTokenId>
    </Token>
  </GetTokensResult>
  <ResponseMetadata>
    <RequestId>c9db3c80-ff03-4a32-b6b6-ee071cd118c8:0</RequestId>
  </ResponseMetadata>
</GetTokensResponse>
```

## Related Actions

- `GetTokenUsage`

# GetTokenUsage

## Description

The `GetTokenUsage` action returns the usage of the given token ID over the last two time periods for the limits defined on the token. You define the limits against which the usage is measured before installing the token with the Amazon FPS Co-Branded service.

### Note

This action works only with multi-use and recurring-use tokens. It does not return token usage for single-use tokens.

## Request Parameters

Parameter	Description	Required
<code>TokenId</code>	The token ID for the token you want usage data for. Type: String Default: None Constraint: Max size = 64 characters	Yes

You must also use parameters that are common to all requests that are described in Common Request Parameters. The common parameters must be explicitly added in REST calls. Parameter names are case sensitive.

## Response Elements

Element	Description
<code>TokenUsageLimits</code>	A list containing the details of this token's usage for each limit defined while installing the token. Type: <code>TokenUsageLimit</code>

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

## Errors

This action can return the following errors:

- `AccessFailure`
- `AuthFailure`
- `InternalServerError`
- `InvalidClientTokenId`
- `InvalidParams`
- `InvalidTokenId`

## Amazon FPS API Reference

- InvalidTokenType
- SignatureDoesNotMatch
- TokenAccessDenied

### Sample REST Request

```
https://fps.amazonaws.com/?
Action=GetTokenUsage
&accessKey=AKIAIOSFODNN7EXAMPLE
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2008-08-06T13%3A00%3A01Z
&TokenId=254656Example83987
&Signature=[URL-encoded signature value]
&Version=2008-09-17
```

### Sample Response to REST Request

```
<GetTokenUsageResponse
  xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <GetTokenUsageResult>
    <TokenUsageLimits>
      <Amount>
        <CurrencyCode>USD</CurrencyCode>
        <Value>10.000000</Value>
      </Amount>
      <LastResetAmount>
        <CurrencyCode>USD</CurrencyCode>
        <Value>0.000000</Value>
      </LastResetAmount>
      <LastResetTimestamp>
        2008-01-01T02:00:00.000-08:00
      </LastResetTimestamp>
    </TokenUsageLimits>
    <TokenUsageLimits>
      <Count>1</Count>
      <LastResetCount>0</LastResetCount>
      <LastResetTimestamp>
        2008-01-01T02:00:00.000-08:00
      </LastResetTimestamp>
    </TokenUsageLimits>
  </GetTokenUsageResult>
  <ResponseMetadata>
    <RequestId>9faeed71-9362-4eb8-9431-b99e92b441ee:0</RequestId>
  </ResponseMetadata>
</GetTokenUsageResponse>
```

### Related Actions

- GetTokens

## GetTransaction

### Description

The GetTransaction action returns details of the transaction specified in transactionId. You can use this action only for transactions within your own account.

### Request Parameters

Parameter	Description	Required
TransactionId	Transaction ID of the transaction you want to get. Type: String Default: None Constraint: Max size = 35 characters	Yes

You must also use parameters that are common to all requests that are described in Common Request Parameters. Parameter names are case sensitive.

### Response Elements

Element	Description
Transaction	Contains the transaction details. Type: TransactionDetail.

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

### Errors

This action can return the following synchronous errors, which occur within the status for this action.

- AccessFailure
- AuthFailure
- InternalError
- InvalidClientTokenId
- InvalidParams
- InvalidTransactionId
- SignatureDoesNotMatch

### Sample REST Request

```
https://fps.sandbox.amazonaws.com/?
Action=GetTransactionsForSubscription
&SubscriptionId=SubscriptionId
&Version=2008-09-17
&AWSAccessKeyId=AccessKey
&Timestamp=2011-03-11T06%3A59%3A40Z
&SignatureVersion=2
```

## Amazon FPS API Reference

```
&Signature=SignatureCalculated  
&SignatureMethod=HmacSHA256
```

### Sample Response to REST Request

```
<GetTransactionResponse xmlns=  
  "http://fps.amazonaws.com/doc/2008-09-17/">  
  <GetTransactionResult>  
    <Transaction>  
      <TransactionId>  
        14GK6BGKA7U6OU6SUTNLBI5SBBV9PGDJ6UL  
      </TransactionId>  
      <CallerReference>CallerReference02</CallerReference>  
      <CallerDescription>MyWish</CallerDescription>  
      <DateReceived>2009-10-05T22:50:08.010-07:00</DateReceived>  
      <DateCompleted>2009-10-05T22:50:09.086-07:00</DateCompleted>  
      <TransactionAmount>  
        <CurrencyCode>USD</CurrencyCode>  
        <Value>1.000000</Value>  
      </TransactionAmount>  
      <FPSFees>  
        <CurrencyCode>USD</CurrencyCode>  
        <Value>0.100000</Value>  
      </FPSFees>  
      <FPSFeesPaidBy>Recipient</FPSFeesPaidBy>  
      <SenderTokenId>  
        553ILMLCG6Z8J431H7BX3UMN3FFQU8VSNTSRNCTAASDJNX66LNZLKSZU3PI7TXIH  
      </SenderTokenId>  
      <FPSOperation>Pay</FPSOperation>  
      <PaymentMethod>CC</PaymentMethod>  
      <TransactionStatus>Success</TransactionStatus>  
      <StatusCode>Success</StatusCode>  
      <StatusMessage>  
        The transaction was successful and the payment instrument  
        was charged.  
      </StatusMessage>  
      <SenderName>Test Business</SenderName>  
      <SenderEmail>new_premium@amazon.com</SenderEmail>  
      <CallerName>Test Business</CallerName>  
      <RecipientName>Test Business</RecipientName>  
      <RecipientEmail>test-caller@amazon.com</RecipientEmail>  
      <StatusHistory>  
        <Date>2009-10-05T22:50:08.092-07:00</Date>  
        <TransactionStatus>Pending</TransactionStatus>  
        <StatusCode>PendingNetworkResponse</StatusCode>  
      </StatusHistory>  
      <StatusHistory>  
        <Date>2009-10-05T22:50:09.086-07:00</Date>  
        <TransactionStatus>Success</TransactionStatus>  
        <StatusCode>Success</StatusCode>  
      </StatusHistory>
```

## Amazon FPS API Reference

```
</Transaction>
</GetTransactionResult>
<ResponseMetadata>
  <RequestId>0702960e-8221-4e04-9413-ca7d010d3b06:0</RequestId>
</ResponseMetadata>
</GetTransactionResponse>
```

### Related Actions

- [GetTokens](#)

## GetTransactionStatus

### Description

The `GetTransactionStatus` action returns the status of the transaction specified by the `TransactionId`.

### 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. For more information, see “Common Request Parameters.”

### Response Elements

Element	Description
<code>CallerReference</code>	A value you provide that uniquely identifies the request. Type: String Size: 128 bytes
<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 <code>PENDING</code> , this field might be <code>PendingVerification</code> , or <code>PendingNetworkResponse</code> . Type: String Size: 64 bytes Valid Values: See “Status Codes”
<code>StatusMessage</code>	A description of the transaction status. Type: String (LOB, Large Object)
<code>TransactionId</code>	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS.

## Amazon FPS API Reference

	Type: String Size: 35 Bytes
TransactionStatus	The status of the transaction. Provides a short code on the status of the transaction, for example "PENDING." Type: TransactionStatus Size: 20 bytes

Responses also include elements common to all responses. For more information, see "Common Response Elements."

### Status Codes

This action can return the following values for StatusCode.

Status Code	Message
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 synchronous errors, which occur within the status for this action.

- AccessFailure
- AuthFailure
- InternalError
- InvalidClientTokenId
- InvalidParams
- InvalidTransactionId
- SignatureDoesNotMatch

### Sample REST Request

```
https://fps.sandbox.amazonaws.com?
```

## Amazon FPS API Reference

```
Action=GetTransactionStatus
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Signature=2l60qD6%2BDIfVEN7ZiHM0AcUKACZt0GYKftIryqkCb6g%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T09%3A12%3A06.921Z
&TransactionId=14GKE3B85HCMF1BTSH5C4PD2IHZL95RJ2LM
&Version=2008-09-17
```

### Sample Query Request

```
GET\n
fps.sandbox.amazonaws.com\n
Action=GetTransactionStatus
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&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>
```

## 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. As of 10 February, 2011, Amazon Payments signs all outbound responses with signature version 2. Unsigned outbound responses are no longer supported.

#### Note

You sign VerifySignature as you would any other Amazon FPS action.

### Request Parameters

Parameter	Description	Required
UrlEndPoint	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 http://my-app-website.biz/, the returnUrl might be http://my-app-website.biz/amazon/success.php, and the IPNUrl might be http://my-app-website.biz/amazon/ipnProcessor.php. Type: String Default: None Constraint: Cannot be null or empty	Yes
HttpParameters	Concatenated string of all URL-Encoded parameters which were included in the response containing the signature you want to verify. This includes the certificateUrl, signatureVersion, signatureMethod and signature parameters. For example, a correctly formatted and URL-encoded string resembles the following:  First%20Name=Joe&Last%20Name=Smith &signatureVersion=2 &signatureMethod=HMACSHA256 &certificateUrl=https%253A%252F%252Ffyps.amazonaws.com%252Fcert%252Fkey.pem &signature=aoeuAOE123eAUdhf]  <b>Tip</b> For validating the returnUrl, you can extract the query string from the returnUrl (excluding the '?' character). For validating the IPNUrl, concatenate the POST parameters.	Yes

## Amazon FPS API Reference

Parameter	Description	Required
	Type: String Default: None Constraint: Cannot be null or empty. In addition, because VerifySignature is a component of signature version 2, the value for signatureVersion must be 2.	

You must also use the Action parameter as described in Common Request Parameters. Parameter names are case sensitive.

### Response Elements

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

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

### Errors

This action can return the following errors:

- InternalServerError
- InvalidParams

### Sample REST Request

This section shows a sample request.

```
https://fps.sandbox.amazonaws.com/?Action=VerifySignature&UrlEndPoint=http%3A%2F%2Fexample.com%3A8080%2Fipn.jsp&HttpParameters=expiry%3D08%252F2015%26signature%3DynDukZ9%252FG77uSJVb5YM0cadwHVwYKPMKOO3PNvgADbv6VtymgBxeOWEhED6KGHSvSjnmWDN%252FZl639AkRe9Ry%252F7zmn9CmiM%252FZkp1XtshERGTqi2YL10GwQpaH17MQqOX3u1cW4LlyFoLy4celUFBPq1WM2ZJnaNZRJIEY%252FvpeVnCVK8VIPdy3HMxPAkNi5zeF2BbqH%252BL2vAWef6vfHkNcJPlOuO16jP4E%252B58F24ni%252B9ek%252FQH1804kw%252FUJ7ZfKwjCCI13%252BcFybpofcKqddq8CuUJj5Ii7Pdw1fje7ktzHeeNhF0r9siWcYmd4JaxTP3NmLJdHFRq2T%252FgsF3vK9m3gw%253D%253D%26signatureVersion%3D2%26signatureMethod%3DRSA-SHA1%26certificateUrl%3Dhttps%253A%252F%252Ffps.sandbox.amazonaws.com%252Fcerts%252F090909%252FPKICert.pem%26tokenId%3DA5BB3HUNAZFJ5CRXIPH72LIODZUNAUZIVP7UB74QNFQDSQ9MN4HPKIKISQZWPLJXF%26status%3DSC%26callerReference%3DcallerReferenceMultiUse1&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Timestamp=2010-02-26T19%3A48%3A05.000Z&Version=2008-09-17&SignatureVersion=2&SignatureMethod=HmacSHA256&Signature=fKRGL42K7nduDA47g6bJCyUyF5ZvkBotXE5jVcgyHvE%3D
```

## Amazon FPS API Reference

### Sample Query Request

```
GET\nfps.sandbox.amazonaws.com\nAction=VerifySignature&UrlEndPoint=http%3A%2F%2Fexample.com%3A8080%2Fipn.jsp&HttpParameters=expiry%3D08%252F2015%26signature%3DynDukZ9%252FG77uSJVb5YM0cadwHVWYKPMK003PNvgADbv6VtymgBxeOWEhED6KGHsGSvSJnMWDN%252FZl639AkRe9Ry%252F7zmn9CmiM%252FZkplXtshERGTqi2YL10GwQpaH17MQqOX3ulcW4LlyFoLy4celUFBPq1WM2ZJnaNZRJIEY%252FvpeVnCVK8VIPdY3HMxPAkNi5zeF2BbqH%252BL2vAWef6vfHkNcJPlOu0l6jp4E%252B58F24ni%252B9ek%252FQH1804kw%252FUJ7ZfKwjCCI13%252BcFybpofcKqddq8CuUJj5Ii7Pdw1fje7ktzHeeNhF0r9siWcYmd4JaxTP3NmLJdHFRq2T%252FgsF3vK9m3gw%253D%253D%26signatureVersion%3D2%26signatureMethod%3DRSA-SHA1%26certificateUrl%3Dhttps%253A%252F%252Ffps.sandbox.amazonaws.com%252Fcerts%252F090909%252FPKICert.pem%26tokenId%3DA5BB3HUNAZFJ5CRXIPH72LIODZUNAUZIVP7UB74QNFQDSQ9MN4HPKIKISQZWPLJXF%26status%3DSC%26callerReference%3DcallerReferenceMultiUse1&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Timestamp=2010-02-26T19%3A48%3A05.000Z&Version=2008-09-17&SignatureVersion=2&SignatureMethod=HmacSHA256&Signature=fKRGL42K7nduDA47g6bJCyUyF5ZvkBotXE5jVcgyHvE%3D
```

### 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>
```

## 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.

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

Parameter	Description	Required
Action	The API operation, for example, Settle or Refund. Type: String Default: None Constraint: Must be a valid operation such as Cancel, Refund, and so on.	Yes

## Amazon FPS API Reference

Parameter	Description	Required
AWSAccessKeyId	A string, distributed by Amazon FPS when you sign up to be a developer, that uniquely identifies the caller. Type: String Default: None	Yes
Signature	A value calculated using the request parameters and a SHA256 (preferred) or SHA1 HMAC encryption algorithm. Type: String Default: None	Yes
SignatureVersion	A value that specifies the Signature format. Type: Integer Default: None Valid Value: 2 Important The previous method for signing (signature version 1) was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. Whenever you sign a request with your access keys, you must now use signature version 2.	Yes
SignatureMethod	A value that specifies the signing method. Type: String Default: None Valid Values: HmacSHA256 (preferred) and HmacSHA1.	Yes
Timestamp	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: dateTime, for example, 2008-09-18T13:00:01Z Default: None	Yes
Version	The version number of the WSDL to use in processing the request. Version numbers are dates, such as 2008-09-17. 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 RequestId element for every API call accepted for processing. The request ID is a reference to your

## Amazon FPS API Reference

	<p>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.</p> <p>Type: String Max Size: 64 Bytes</p>
signatureVersion	<p>A value that specifies the Signature format.</p> <p>Type: Integer Valid Values: 2</p>
signatureMethod	<p>A value that specifies the signing method.</p> <p>Type: String Valid Values: HmacSHA256 (preferred) and HmacSHA1.</p>

## Errors

Error	Description
AccessFailure	<p>Account cannot be accessed.</p> <p>You can display the following message to your customers:</p> <p>Your account cannot be accessed. Retriable: Yes</p>
AccountClosed	<p>Account is not active.</p> <p>You can display the following message to your customers:</p> <p>Your account is closed. Retriable: Yes</p>
AccountLimitsExceeded	<p>The spending or receiving limit on the account is exceeded. This error can also occur when the specified bank account has not yet been verified.</p> <p>You can display the following message to your customers: You have exceeded your spending or receiving limits. You can view your current limits at <a href="http://payments.amazon.com/sdui/sdui/viewlimits">http://payments.amazon.com/sdui/sdui/viewlimits</a>. You can upgrade these limits by adding and verifying a bank account as a payment method. Please visit <a href="#">Adding and Verifying Bank Accounts</a> to learn how to add and instantly verify a bank</p>

## Amazon FPS API Reference

Error	Description
	<p>account. Retriable: Yes</p>
AmountOutOfRange	<p>The transaction amount is more than the allowed range.</p> <p>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</p>
AuthFailure	<p>AWS was not able to validate the provided access credentials.</p> <p>Please make sure that your AWS developer account is signed up for FPS. Retriable: Yes</p>
ConcurrentModification	<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>
DuplicateRequest	<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>
InactiveInstrument	<p>Payment instrument is inactive.</p> <p>The payment instrument is inactive, for example, a credit card has expired. Retriable: No</p>
IncompatibleTokens	<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> <p>One or more tokens has expired. The recipient specified in the token is different from the actual recipient in the transaction.</p>

## Amazon FPS API Reference

Error	Description
	<p>There is violation on the amount restriction. This token cannot be used with your application as another application has installed it.</p>
InsufficientBalance	<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 Amazon FPS fees does not have a sufficient account balance.</p> <p>Retriable: Yes</p>
InternalError	<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>
InvalidAccountState	<p>The account is either suspended or closed. Payment instructions cannot be installed on this account.</p> <p>You must ask your customer to set up a new account if the account is closed.</p> <p>Retriable: Yes</p>
InvalidAccountState_Caller	<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>
InvalidAccountState_Recipient	<p>Recipient account cannot participate in the transaction.</p> <p>You can display the following message to your customer (sender): Your Amazon Payments account is not active. Please visit <a href="http://payments.amazon.com">http://payments.amazon.com</a> for more details.</p> <p>Retriable: Yes</p>
InvalidAccountState_Sender	<p>Sender account cannot participate in the transaction.</p> <p>You can display the following message to your customer (sender): Your Amazon Payments account is not active. Please visit <a href="http://payments.amazon.com">http://payments.amazon.com</a> for more details.</p>

## Amazon FPS API Reference

Error	Description
	Retriable: Yes
InvalidCallerReference	The Caller Reference does not have a token associated with it. Use the caller reference value that was passed to the InstallPaymentInstruction operation or the Amazon FPS Co-Branded UI pipeline.
InvalidClientTokenId	The AWS Access Key Id you provided does not exist in our records.  Please check that the AWS Access Key Id used to make the request is valid. Retriable: No
InvalidDateRange	The end date specified is before the start date or the start date is in the future. Specify the correct end date.
InvalidParams	One or more parameters in the request is invalid.  For more information, see the parameter descriptions for the action in the <a href="#">FPS API Reference</a> . Parameters are case sensitive. Retriable: No
InvalidPaymentInstrument	The payment method used in the transaction is invalid. Specify a valid payment method
InvalidPaymentMethod	The cause for this error is dependent on the calling action:  For InstallPaymentInstruction, payment method specified in the GK construct is invalid. Specify the correct payment method.
InvalidRecipientForCCTransaction	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.
InvalidSenderRoleForAccountType	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
InvalidTokenId	You did not install the token that you are trying to cancel.

## Amazon FPS API Reference

Error	Description
	<p>You do not have permission to cancel this token. You can cancel only the tokens that you own.</p> <p>Retriable: No</p>
InvalidTokenId_Recipient	<p>The recipient token specified is either invalid or canceled.</p> <p>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.</p> <p>Retriable: No</p>
InvalidTokenId_Sender	<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>
InvalidTokenType	<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>
InvalidTransactionId	<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>
InvalidTransactionState	<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>
NotMarketplaceApp	<p>This is not a 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>
OriginalTransactionFailed	<p>The original transaction has failed.</p> <p>You cannot refund a transaction that has originally failed.</p> <p>Retriable: No</p>
OriginalTransactionIncomplete	<p>The original transaction is still in progress.</p>

## Amazon FPS API Reference

Error	Description
	<p>Retry after the original transaction has completed. Retriable: Yes</p>
PaymentInstrumentNotCC	<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>
PaymentMethodNotDefined	<p>Payment method is not defined in the transaction. Specify the payment method in the sender token.</p>
RefundAmountExceeded	<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>
SameSenderAndRecipient	<p>The sender and receiver are identical, which is not allowed. Retriable: No</p>
SameTokenIdUsedMultipleTimes	<p>This token is already used in earlier transactions. The tokens used in a transaction should be unique.</p>
SenderNotOriginalRecipient	<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>
SettleAmountGreaterThanDebt	<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>
SettleAmountGreaterThanReserveAmount	<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>
SignatureDoesNotMatch	<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</a></p>

## Amazon FPS API Reference

Error	Description
	<a href="#">Service Getting Started Guide</a> . Retriable: No
TokenAccessDenied	Permission is denied to cancel the token. You are not allowed to cancel this token. Retriable: No
TokenNotActive	The token is canceled.  A new token needs to be created. Retriable: No
TokenNotActive_Recipient	The recipient token is canceled.  If you are the recipient, set up a new recipient token using the InstallPaymentInstruction operation or direct your customers to the Recipient Token Installation Pipeline to set up recipient token. Retriable: No
TokenNotActive_Sender	The sender token is canceled.  You must ask your customer to set up a new payment authorization because the current authorization is not active. Retriable: No
TokenUsageError	The token usage limit is exceeded. 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.
TransactionDenied	This transaction is not allowed.  You are not allowed to do this transaction. Check your credentials. Retriable: No
TransactionFullyRefunded Already	This transaction has already been completely refunded.  You are not allowed to refund more than the original transaction amount. Retriable: No
TransactionTypeNotRefundable	You cannot refund this transaction.  Refund is allowed only on the Pay operation. Retriable: No
UnverifiedAccount_Recipient	The recipient's account must have a verified bank account or a credit card before this transaction can be initiated.

## Amazon FPS API Reference

Error	Description
	<p>You can display the following message to your customer (recipient): Your Amazon Payments account is not active. Please visit <a href="http://payments.amazon.com">http://payments.amazon.com</a> for more details.</p> <p>Retriable: No</p>
UnverifiedAccount_Sender	<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: Please add a U.S. credit card or U.S. bank account and verify your bank account before making this payment.</p> <p>Retriable: No</p>
UnverifiedBankAccount	<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>
UnverifiedEmailAddress_Caller	<p>The caller account must have a verified email address.</p> <p>You cannot make a web service API call without verifying your email address. Go to <a href="http://payments.amazon.com">http://payments.amazon.com</a> web site and make payments.</p> <p>Retriable: No</p>
UnverifiedEmailAddress_Recipient	<p>The recipient account must have a verified email address for receiving payments.</p> <p>You can display the following message to your customers: You cannot receive payments. Please verify your email address. Go to <a href="http://payments.amazon.com">http://payments.amazon.com</a> to verify your account and receive payments.</p> <p>Retriable: No</p>
UnverifiedEmailAddress_Sender	<p>The sender account must have a verified email address for this payment</p> <p>You can display the following message to your customers: You cannot make payments. Please verify your email address. Go to</p>

## Amazon FPS API Reference

Error	Description
	<a href="http://payments.amazon.com">http://payments.amazon.com</a> to verify your account and make payments. Retriable: No

## Data Types

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

## Enumerated Data Types

Topics

- AccountBalance
- ChargeFeeTo
- CurrencyCode
- FPSOperation
- InstrumentId
- InstrumentStatus
- PaymentMethod
- RelationType
- SortOrderByDate
- TokenStatus
- TokenType
- TransactionalRole
- TransactionStatus

This section describes the enumerated data types Amazon FPS uses.

### AccountBalance

Name	Description	Type
AvailableBalances	The total amount of money that is transferred to your account from a bank account transfer or a refund.	AvailableBalances
PendingInBalance	The total amount that is yet to be credited to your account.	Amount
PendingOutBalance	The total amount that is yet to be debited from your account.	Amount
TotalBalance	The total balance that is currently available in your account.	Amount

### ChargeFeeTo

Name	Description	Type
Caller	Caller shall pay the fees.	String
Recipient	Recipient shall pay the fees.	String

### CurrencyCode

Name	Description	Type
USD	The transaction uses U.S. dollars.	String

## Amazon FPS API Reference

### FPSOperation

These values are returned for non-IPN operations.

Name	Description	Type
Pay	All pay transactions.	String
Refund	All refund transactions.	String
Settle	All settle transactions.	String
Reserve	All reserve transactions.	String

These values are returned only for IPN operations.

Name	Description	Type
PAY	All pay transactions.	String
REFUND	All refund transactions.	String
SETTLE	All settle transactions.	String
RESERVE	All reserve transactions.	String
MULTI_SETTLE	All multi-settle transactions.	String
REAUTH	All transactions that required reauthorization.	String
DEPOSIT_FUNDS	All fund deposit transactions.	String
WITHDRAW_FUNDS	All fund withdrawal transactions.	String
CANCEL_TRANSACTION	All non-user cancelled transactions.	String
CANCEL	All non-user cancelled String transactions.	String

### InstrumentId

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

### InstrumentStatus

Name	Description	Type
Active	All active instruments installed for your application.	String
All	All instruments installed for your application.	String
Cancelled	All canceled instruments.	String

### PaymentMethod

Name	Description	Type
ABT	Amazon Payments account balance transfer.	String
ACH	Bank account transaction.	String

## Amazon FPS API Reference

CC	Credit card transaction.	String
----	--------------------------	--------

### RelationType

Name	Description	Type
MarketplaceFee	Marketplace fee transactions.	String
Parent	Parent transactions.	String
Refund	Refund transactions.	String
RefundReversal	RefundReversal transactions.	String
Reserve	Reserve transactions.	String
Settle	Settle transactions.	String

### SortOrderByDate

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

### TokenStatus

Name	Description	Type
Active	The token is in active state.	String
Inactive	The token was canceled by the user and is inactive.	String

### TokenType

Name	Description	Type
MultiUse	Token that can be used multiple times.	String
Recurring	Token which is specifically marked for recurring payments.	String
SingleUse	Token that can be used only once.	String
Unrestricted	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
Caller	Role is the caller.	String
Recipient	Role is the recipient.	String
Sender	Role is the sender.	String

### TransactionStatus

These values are returned for non-IPN operations.

Name	Description	Type
------	-------------	------

## Amazon FPS API Reference

Cancelled	The transaction was canceled.	String
Failure	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
Pending	The transaction is pending.	String
Reserved	The reserve request on the transaction succeeded. Amazon FPS reserves the purchase price against the sender's payment instrument.	String
Success	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
CANCELLED	The transaction was canceled.	String
FAILURE	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
PENDING	The transaction is pending.	String
RESERVED	The reserve request on the transaction succeeded. Amazon FPS reserves the purchase price against the sender's payment instrument.	String
SUCCESS	The transaction succeeded. You can fulfill the order for the customer.	String

## Complex Data Types

This section describes the complex data types Amazon FPS uses.

### Amount

Name	Description	Type
CurrencyCode	The currency code of the amount. Amazon FPS currently supports only USD.	CurrencyCode
Value	The numeric value of the amount in dollars. Two optional decimal places are allowed. For example, 25.01 is \$25.01, and 2500 is \$2500.	String

### AvailableBalances

Name	Description	Type
DisburseBalance	The total balance that has been disbursed.	Amount
RefundBalance	The total amount that has been refunded.	Amount

### DebtBalance

Name	Description	Type
AvailableBalance	Available debt balance accumulated between recipient and sender.	Amount
PendingOutBalance	Any balance that is pending because of an external instrument was used to settle the debt.	Amount

### DescriptorPolicy

Name	Description	Type
CSOwner	The recipient or caller customer service number. If you specify Caller, 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 CSOwner is Recipient.	The entity whose CS Phone number should be used. Valid values are either Recipient or Caller. Default: Recipient
SoftDescriptorType	The type of soft descriptor. Valid values are either Static or Dynamic. If you specify Static, or do not specify a type, the soft descriptor in your account level setting is sent to the payment processor. If you specify Dynamic, the first 15 characters of sender description	The type of soft descriptor. Valid values are either Static or Dynamic. Default: Static

## Amazon FPS API Reference

Name	Description	Type
	is sent to the payment processor.	

### MarketplaceRefundPolicy

Name	Description	Type
MarketplaceTxnOnly	Caller refunds his fee to the recipient.	String
MasterAndMarketplaceTxn	Caller and Amazon FPS refund their fees to the sender, and the recipient refunds his amount	String
MasterTxnOnly	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

### RelatedTransaction

Name	Description	Type
RelationType	Relation type of the related transaction.	RelationType
TransactionId	The Transaction ID of the related transaction.	String Max size = 35 characters

### StatusHistory

Name	Description	Type
Amount	The changed amount.	Amount
Date	The date when the status changed.	dateTime
StatusCode	The current status of the transaction.	String
TransactionStatus	The current status of the transaction.	TransactionStatus

### Token

Name	Description	Type
CallerReference	Account ID of the caller who initiated the original request.	String Max size = 128 bytes
DateInstalled	The date and time when the payment token was created on the caller's account.	dateTime
FriendlyName	A name that references the token.	String Max size = 128 characters
OldTokenId	The token ID linked to this token. The token that was created in place of this token.	String Size: 65 Bytes
PaymentReason	Payment reason passed during token installation.	String
TokenId	The token ID representing the payment	String

## Amazon FPS API Reference

Name	Description	Type
	instruction.	Max size = 64 characters
TokenStatus	Specifies whether or not the token is active.	TokenStatus
TokenType	The type of the token (e.g., single-use, multi-use, etc.).	TokenType

### TokenUsageLimit

Name	Description	Type
Amount	Amount paid in the latest time window with this token.	Amount
Count	Number of times this token was used in the latest time window.	Integer
LastResetAmount	Amount paid in the previous time window with this token.	Amount
LastResetCount	Number of times this token was used in the previous time window.	Integer
LastResetTimeStamp	The exact time when the latest time window started for this limit.	dateTime

### Transaction

Name	Description	Type
CallerName	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
CallerTransactionDate	Date the caller provided for the transaction.	dateTime
DateCompleted	Date the transaction was completed.	dateTime
DateReceived	Date the transaction was received by Amazon FPS.	dateTime
FPSFees	Amount of fees collected by Amazon FPS for performing the transaction.	Amount
FPSOperation	The operation type.	FPS Operation
OriginalTransactionId	In the case of a refund, the TransactionID that is being reversed.	String Max size = 35 characters
PaymentMethod	Payment method used in the transaction.	Payment Method
RecipientName	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

## Amazon FPS API Reference

Name	Description	Type
RecipientTokenID	The recipient token used in the transaction. Recipient tokens are needed when the caller and recipient are different people.	String Size: 65 Bytes
SenderName	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
SenderTokenID	The sender token used in the transaction.	String Size: 65 Bytes
StatusCode	A code that represents the current status of the transaction. Expands on the information in the TransactionStatus field. For example, if TransactionStatus is PENDING, this field might be PendingVerification, or PendingNetworkResponse.	String
StatusMessage	A short description of the current status of the transaction.	String
TransactionAmount	Total amount of the transaction.	Amount
TransactionId	Unique Amazon FPS-generated ID for the transaction.	String Max size = 35 characters
TransactionPart	List of individual parts of the transaction, with each one dealing with your account's role in the transaction.	Transaction Part
TransactionStatus	Provides a short code on the status of the transaction, for example "PENDING."	Transaction Status

### TransactionDetail

Name	Description	Type
CallerNamePDF	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
CallerDescription	Caller description the caller provided for the transaction.	String Constraint: Max size = 160 characters
CallerReference	Caller reference the caller provided for the transaction.	String Max size = 128

## Amazon FPS API Reference

Name	Description	Type
		characters
DateReceived	Date Amazon FPS received the transaction.	dateTime
DateCompleted	Date the transaction was completed.	dateTime
FPSFees	Amount of fees collected by Amazon FPS for performing the transaction.	Amount
FPSFeesPaidBy	The party paying the FPS fees for this transaction.	TransactionalRole
FPSOperation	The operation type.	FPSOperation
MarketPlaceFees	In the case of a marketplace transaction, this is the amount of any marketplace fee the caller has charged.	Amount
PaymentMethod	The payment method used.	PaymentMethod
RecipientEmail	The email ID of the recipient of this transaction.	String
RecipientName	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
RecipientTokenId	Recipient token ID used in the transaction. Recipient tokens are needed when the caller and recipient are different people.	String Size: 65 Bytes
RelatedTransaction	All transactions related to this transaction.	RelatedTransaction
SenderDescription	Sender description the caller provided for the transaction.	String Constraint: Max size = 160 characters
SenderEmail	The email ID of the sender of this transaction. This is returned only if the caller is also the recipient of this transaction.	String
SenderName	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
SenderTokenId	Sender token ID used in the transaction.	String Size: 65 Bytes
StatusCode	A code that represents the current status of the transaction.	String
StatusHistory	A list of all the previous status entries for this transaction.	StatusHistory
StatusMessage	A short description of the current status of the transaction.	String

## Amazon FPS API Reference

Name	Description	Type
TransactionAmount	Total amount of the transaction.	Amount
TransactionId	Unique Amazon FPS-generated ID for the transaction.	String Max size = 35 characters
TransactionStatus	The transaction status.	TransactionStatus

### TransactionPart

Name	Description	Type
Description	Description provided by the entity.	String
FeesPaid	Fees the caller or recipient paid.	Amount
InstrumentId	Payment instrument involved in this transaction part.	String
Name	Name used for the role specified in Role.	String
Reference	Reference data provided by this party.	String
Role	Role played by this party.	TransactionalRole

# Code Samples

The following sections provide information about the Amazon Flexible Payments Service (FPS) development libraries and sample code 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="#">Amazon FPS Quick Starts (API version - 2010-08-28): Standalone Library in C#</a>
Java	<a href="#">Amazon FPS Quick Starts (API version - 2010-08-28): Standalone Library in Java</a>
Perl	<a href="#">Amazon FPS Quick Starts (API version - 2010-08-28): Standalone Library in Perl</a>
PHP	<a href="#">Amazon FPS Quick Starts (API version - 2010-08-28): Standalone Library in PHP</a>

Each package is updated for signature version 2, and contains both a development library and a collection of 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 quick start
- Generate signatures compliant with signature version 2
- Validate the content of return URL responses and IPN notifications

For 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

## Code Samples

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-dependent 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 CBUI and Return URL/IPN Validations samples are different. For more information, “Understanding the Amazon CBUI Samples.”)

## 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 identically 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.”

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 [package-root]/src and [package-root]/third-party folder are in the java classpath.
2. In the [package-root]/src/config.properties file, set the values for AwsAccessKey and AwsSecretKey using your security credentials.

## Code Samples

### Important

Your Secret Access Key is a secret, which only you and Amazon should know. It is important to keep it confidential to protect your account. Store it securely. Never include it in your requests to the Amazon Flexible Payments Service (Amazon FPS), and never email it to anyone. Do not share it outside your organization, even if an inquiry appears to come from Amazon Web Services (AWS) or Amazon.com. No one who legitimately represents Amazon will ever ask you for your Secret Access Key.

To get your credentials, see “Getting an AWS Account” in the [Amazon Simple Pay Getting Started Guide](#).

In the same file, if you want to target the sandbox, change the `AwsServiceEndPoint` property to <https://fps.sandbox.amazonaws.com>. Then save the file.

3. In the `[package-root]/src/com/amazonaws/fps/samples/VerifySignature.java` file, find the section containing the lines:

```
VerifySignatureRequest fpsRequest = new
    VerifySignatureRequest();
// @TODO: set request parameters here
// invokeVerifySignature(service, fpsRequest);
```

(The `VerifySignatureRequest`, `VerifySignatureResult`, and `VerifySignatureResponse` classes are located in

`[package-root]/src/com/amazonaws/fps/model` folder.)

4. In the same file, remove the comment on `invokeVerifySignature`, and after it add the `VerifySignature` parameter assignments consistent with your transaction. For example:

```
fpsRequest.setAction("VerifySignature");
fpsRequest.setUrlEndpoint("http://myApplication/my-ipn-
response.pgp");
fpsRequest.setHttpParameters(
    "Name1=Joe&
    "Name2=College&" +
    "signatureVersion=2&" +
    "signatureMethod=HMACSHA256&" +
    "certificateUrl=https://fps.amazonaws.com/cert/key.pem&" +
    "signature=aoeuAOE123eAUdhf");
```

Save the file. For information on the parameters to `VerifySignature`, see “VerifySignature.”

5. Compile and run the sample.

The program copies to standard out a representation of the `VerifySignatureResponse` XML fragment similar to the following:

```
VerifySignature Action Response
=====
VerifySignatureResponse
```

## Code Samples

```
VerifySignatureResult
  True
VerificationStatus
  Success
ResponseMetadata
  RequestId
    bda6-4f5f-b37b-1a146b9a-b9e45c3012a5:0
```

For information on the XML document returned by `VerifySignature`, see “`VerifySignature`.”

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.

### 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>

**Note**  
The Visual Studio.NET samples are organized into this solution. After setting your access parameters the first time, you

## Code Samples

File	Location
build the entire solution to generate the dependency classes. Then you modify the specific sample you want. See the Readme.html file for more information.	

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

File	Location
VerifySignatureRequest.pm	[package root]/src/Amazon/FPS/Model
VerifySignatureSample.pl	[package root]/src/Amazon/FPS/Samples
ReadMe.html Readme for perl fps library	[package root]/src

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

File	Location
VerifySignatureRequest.php	[package root]/src/Amazon/FPS/Model
VerifySignatureSample.php	[package root]/src/Amazon/FPS/Samples
ReadMe.html Readme for php fps library	[package root]/src

# 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
CBUI Single Use Pipeline Sample	Requests authorization for a one-time payment.
CBUI Multi-Use Pipeline Sample	Requests authorization for multiple payments.
CBUI Recipient Pipeline Sample	Requests authorization for a recipient token pipeline, such as that needed for marketplace fixed and variable fees.
CBUI Recurring Token Pipeline Sample	Requests authorization for a recurring token pipeline, such as that needed for periodic charges.
CBUI Edit Token Pipeline Sample	Requests authorization for an edit-token pipeline.

In the following section, we show how to work with the CBUI Single Use Pipeline 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 CBUI Single Use Pipeline sample is the same for all the other samples in the `CBUI/Samples` (or, in the case of `Amazon.CBUI.Samples`)

## Code Samples

folder. (The process for the FPS and Return URL/IPN Validations samples are different. For more information, see “Understanding the Amazon FPS Samples.”)

## 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.

The CBUISingleUsePipeline sample centers on the following files:

File	Description
config.properties	<p>Set your AWS access key ID, AWS secret key, and sandbox endpoint in this file.</p> <p><b>Important</b> Your Secret Access Key is a secret, which only you and Amazon should know. It is important to keep it confidential to protect your account. Store it securely. Never include it in your requests to the Amazon Flexible Payments Service (Amazon FPS), and never email it to anyone. Do not share it outside your organization, even if an inquiry appears to come from Amazon Web Services (AWS) or Amazon.com. No one who legitimately represents Amazon will ever ask you for your Secret Access Key.</p>
CBUISingleUsePipelineSample.java	<p>In the main method, you create an AmazonFPSSingleUsePipeline object and use it to add parameters specific to your application.</p>
AmazonFPSSingleUsePipeline.java	<p>Invoked from CBUISingleUsePipelineSample.java, this class contains the setMandatoryParameters and validateParameters functions which you can customize for your application.</p>

## Code Samples

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

1. Open the file [package-root]/src/config.properties, and set AwsAccessKey and AwsSecretKey properties to your AWS access key and AWS secret key, respectively.

#### Important

Your Secret Access Key is a secret, which only you and Amazon should know. It is important to keep it confidential to protect your account. Store it securely. Never include it in your requests to the Amazon Flexible Payments Service (Amazon FPS), and never email it to anyone. Do not share it outside your organization, even if an inquiry appears to come from Amazon Web Services (AWS) or Amazon.com. No one who legitimately represents Amazon will ever ask you for your Secret Access Key.

To get your security credentials, see “Getting an AWS Account” in the [Amazon Flexible Payments Service Getting Started Guide](#).

2. In the same file, set the AwsServiceEndPoint to `https://fps.sandbox.amazonaws.com/` (the Amazon FPS sandbox).
3. In the same file, set the CBUIServiceEndPoint to `https://authorize.payments-sandbox.amazon.com/cobranded-ui/actions/start` (the Co-Branded service sandbox).
4. Open the file [package-root]/src/com/amazonaws/cbui/samples/CBUISingleUsePipelineSample.java, and find the following line:

```
AmazonFPSSingleUsePipeline pipeline= new  
AmazonFPSSingleUsePipeline(accessKey, secretKey);
```

Change the pipeline.setMandatoryParameters and pipeline.addParameters method calls to the following:

```
//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">;  
pipeline.addParameter("callerReference", "[Unique ID for the  
transaction]");
```

Save the file.

5. Ensure that all the jar files in the third-party 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. Because the sender and recipient cannot be the same, you must use an account different from your AWS developer or business accounts.

## Code Samples

- When complete, the page you specified as [your returnUrl] is hit with the Co-Branded service response. You validate this response by testing the signature.

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 validateParametersfunction 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.

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

File	Location
CBUISingleUsePipelineSample.cs	[package root]\src\Amazon.CBUI\Amazon.CBUI.Model.
AmazonFPSSingleUsePipeline.cs	[package root]\src\Amazon.CBUI.Samples\Amazon.CBUI.Samples
Amazon.CBUI.proj Visual Studio.NET solution for the development library	[package root]\src\Amazon.CBUI\Amazon.CBUI\
Amazon.CBUI.Samples.proj Visual Studio.NET solution for the Amazon FPS API samples	[package root]\src\Amazon.CBUI\Amazon.CBUI.Samples\Amazon.CBUI.Samples
Amazon.FPS.sln Visual Studio.NET solution for the library package	[package root]/src/Amazon.FPS/Amazon.FPS

**Note**  
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 Readme.html file for more information.

## Code Samples

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

File	Location
CBUISingleUsePipelineSample.pl	[package root]/src/Amazon/CBUI/Samples.
AmazonFPSSingleUsePipeline.pm	[package root]/src/Amazon/CBUI
ReadMe.html readme for perl fps library	[package root]/src

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

File	Location
CBUISingleUsePipelineSample.php	[package root]/src/Amazon/CBUI/Model
CBUISingleUsePipeline.php	[package root]/src/Amazon/CBUI/Samples
ReadMe.html readme for php fps library	[package root]/src

## Understanding the IPNAndReturnURLValidation Sample

Amazon provides samples in four programming languages which show you how to perform a server-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.

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

File	Description
ReturnUrlVerificationSampleCode.java	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.
IPNVerificationSampleCode.java	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.
SignatureUtilsForOutbound.java	Invoked from <code>ReturnUrlVerificationSampleCode.java</code> and <code>IPNVerificationSampleCode.java</code> ,

## Code Samples

File	Description
	this class uses the signature version 2 process to validate the signature. 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 IPNAndReturnURLValidation 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 `src/com/amazonaws/ipnreturnurlvalidation` 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 `ReturnUrlVerificationSampleCode` and `IPNVerificationSampleCode` classes use a `HashMap` 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.

3. Compile the sample. For example, if you are including the `[package-root]` `src/third-party/commons-codec-1.3/commons-codec-1.3.jar` using the linux command line, you would type

```
$javac -cp .:[package-root]  
src/third-party/commons-codec-1.3/commons-codec-1.3.jar  
ReturnUrlVerificationSampleCode.java  
SignatureUtilsForOutbound.java
```

On Windows, you would type

```
$javac -cp .:[package-root]  
src/third-party/commons-codec-1.3/commons-codec-1.3.jar  
ReturnUrlVerificationSampleCode.java  
SignatureUtilsForOutbound.java
```

4. Run the sample. Continuing the previous example, on linux, you would type

```
$javac -cp .:[package-root]  
src/third-party/commons-codec-1.3/commons-codec-1.3.jar  
ReturnUrlVerificationSampleCode
```

## Code Samples

On Windows, you would type

```
$javac -cp .:[package-root]  
src/third-party/commons-codec-1.3/commons-codec-1.3.jar  
ReturnUrlVerificationSampleCode
```

The result "**Is signature correct: true**" is printed to standard out if the verification determines the signature to be valid.

## 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
ReturnUrlVerificationSampleCode.cs	[package root] src\ Amazon.IpnReturnUrlValidation\.
IPNVerificationSampleCode.cs	[package root] src\Amazon.IpnReturnUrlValidationSamples. IpnReturnUrlValidationSamples\
SignatureUtilsForOutbound.cs	[package root] src\Amazon.IpnReturnUrlValidationSamples. IpnReturnUrlValidationSamples\
IpnAndReturnUrlValidation.html Readme for this sample	[package root] src\Amazon.IpnReturnUrlValidationSamples. IpnReturnUrlValidationSamples\
IpnReturnUrlValidation.Samples.csproj Visual Studio.NET project for this sample	[package root] src\Amazon.IpnReturnUrlValidation\

### Perl File Locations for the IpnReturnUrlValidation Library

Class	Location
ReturnUrlVerificationSampleCode.pl	[package root] src/Amazon/ IpnReturnUrlValidation/Samples.
IPNVerificationSampleCode.pl	[package root] src/Amazon/ IpnReturnUrlValidation/Samples
SignatureUtilsForOutbound.pm	[package root] src/Amazon/ IpnReturnUrlValidation
IpnAndReturnUrlValidation.html Readme for this sample	[package root] src

### PHP File Locations for the IpnReturnUrlValidation Library

Class	Location
-------	----------

## Code Samples

<code>ReturnUrlVerificationSampleCode.pl</code>	<code>[package root] src/Amazon/IpnReturnUrlValidation /Samples..</code>
<code>IPNVerificationSampleCode.pl</code>	<code>[package root] src/Amazon/IpnReturnUrlValidation /Samples</code>
<code>SignatureUtilsForOutbound.pm</code>	<code>[package root] src/Amazon/IpnReturnUrlValidation</code>
<code>IpnAndReturnUrlValidation.html</code> 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.jspx>. The **Discussion Forums** page opens.
2. From the **Developers** 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."

# 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>	Gets you set up with Amazon FPS, and shows you how to implement a simple one-time payment using Amazon FPS Basic Quick Start.
<a href="#">Amazon Flexible Payments Service Marketplace Quick Start</a>	Covers the marketplace functionality of Amazon FPS.
<a href="#">Amazon Flexible Payments Service Advanced Quick Start</a>	Covers the multiple-payment functionality of Amazon FPS.
<a href="#">Amazon Flexible Payments Service Account Management Quick Start</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>	Provides a high-level overview of the current release, noting any new features, corrections, and known issues.
<a href="#">FPS Developer Resource Center</a>	A starting point specifically for FPS documentation, code samples, release notes, and other information to help you build innovative applications.
<a href="#">Discussion Forums</a>	A community-based forum for developers to discuss technical questions related to Amazon FPS.
<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, accounts, events, abuse, and more.
<a href="#">Conditions of Use</a>	Detailed information about Amazon.com copyright and trademark usage and other topics.

## Glossary

# Glossary

ABT, Amazon Payments Account Balance	One method of payment available with Amazon Payments.
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.
ACH	Bank Account Debits One method of payment available with Amazon Payments.
buyer	The buyer pays the seller for a product or service.
caller	A developer who facilitates payment between a sender and a recipient.
chargeback	a reversal of a payment issued by the bank when the buyer disputes the charge.
Co-Branded User Interface (CBUI)	A set of Amazon Payments web pages which lead a user through a secure login and payment authorization pipeline. Once the pipeline is complete, the user is redirected to your website.
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 Secret Key. 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, marketplace scenario	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.

## Glossary

outbound notifications	Response from Amazon Payments to your Amazon FPS application by way of Return URL or IPN.
payment instrument	The method of payment a customer chooses to use with Amazon Payments. These are credit cards, Amazon Payments account balance (ABT), and bank account debits (ACH).
one time payment	An Amazon FPS payment processed with a single-use payment token. When the payment is made, the token may no longer be used.
recipient	A seller who receives a payment from a buyer (sender) in exchange for a service or product.
Recipient Token	Payment token created when a seller authorizes a payment of marketplace fees to you for hosting services, often with a Register Now button.
recurring payment	An Amazon FPS payment processed with a recurring payment token. Payments are made periodically using the same payment token. The token is valid until it expires.
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	The seller receives money from a buyer in exchange for a service or product.
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

## Glossary

	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.
Sender Token	Payment token created when buyers authorize purchase on their own behalf, often with a Pay Now button.
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.
website owner	A developer who uses Amazon Flexible Payments Service.

# Document History

This documentation is associated with the 2010-08-28 version of the *Amazon FPS Account Management Quick Start*. This guide was last updated on 10-December-2012.

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

Change	Description	Release Date
Enhancement	Added minor changes and typographical fixes applied from a maintenance edit.	In this release
Feature Deprecation	Amazon FPS has removed support for signature verification using signature version 1. If your application is using this feature, you must convert to a server-side call with Verifying the ReturnURL and IPN Notifications.	2011-02-10
Feature Deprecation	Amazon FPS has removed support for client-side signature verification using PKI. If your application is using this feature, you must convert to a server-side call with Verifying the ReturnURL and IPN Notifications.	2011-02-10
Maintenance Update	Added an expanded breadcrumb to the HTML version, and reduced the front matter content. In addition several minor fixes and corrections have been applied.	2010-11-01
Enhancement	The RecipientVerificationStatus data type (used by the GetRecipientVerificationStatus action) has been enhanced to enable you to determine whether a verified customer account is unlimited in the amount of money it can receive. For more information, see the <a href="#">Amazon Flexible Payments Service Advanced Quick Start</a> or the <a href="#">Amazon Flexible Payments Service Marketplace Quick Start</a> .	2010-11-01
Feature Extension	Amazon FPS has extended support for signature verification using signature version 1 to 10 February 2011. If your application is using this feature, you must convert to a server-side call with Verifying the ReturnURL and IPN Notifications.	2010-11-01
Feature Extension	Amazon FPS has extended support for client-side signature verification using PKI until 10 February 2011. If your application is using this feature, you must convert to a server-side call with Verifying the ReturnURL and IPN Notifications.	2010-09-14
Enhancement	Changed the values for soft descriptors from "AMZ*" and "AMZN PMNTS" to "ASI*" and "Amazon Payments", respectively. Also, applied	2010-06-11

## Document History

Change	Description	Release Date
	minor changes and typographical fixes applied from a maintenance edit.	
Enhancement	Examples of the email messages sent by Amazon Payments which are relevant to Amazon FPS are now included as part of this guide. Please see “Email Notification Templates.”	2010-01-29
New Feature	<p>Support for signature version 2, which completely replaced signature version 1 on 10 February, 2011. The enhanced security features include:</p> <ul style="list-style-type: none"> <li>• a more secure way of calculating signatures for inbound requests and outbound notifications. For more information, see “Working with Signatures.”</li> <li>• support for SHA256 signing algorithm</li> <li>• the new VerifySignature FPS Action for server-side testing of return URL responses and IPN notifications. For more information, see “VerifySignature.”</li> <li>• support for PKI based authentication for client-side testing of return URL responses and IPN notification.</li> </ul> <p><b>Note</b> This feature is deprecated as of 2010-09-14.</p>	2009-11-03
Enhancement	The Access Keys page has been renamed the Security Credentials page, located at <a href="http://aws.amazon.com/security-credentials">http://aws.amazon.com/security-credentials</a> . You must be logged in to view this page.	2009-09-09
Feature Deprecation	Amazon FPS has removed the Aggregated Payments option. References to the AWS Developer Resource Center and AWS Support Center have been removed.	2013-09-13
Editorial Update	Added language to clarify that the Amazon Payments service has been designed and developed for use within a web browser only. Our service cannot be used within a native application (including, without limitation, iOS, Android, RIM and Windows operating systems).	2013-10-18
Correction	Fixed a typo in the description for the unverifiedEmailAddress_Sender error response.	2013-12-06