
Amazon DevPay

Getting Started Guide

Version 2007-12-01



Amazon Web Services

Amazon DevPay: Getting Started Guide

Amazon Web Services

Copyright © 2013 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, Cloudfront, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC. 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.

Welcome	1
What Is Amazon DevPay?	3
The Steps to Using DevPay	6
Amazon S3 Example	12
About the Sample Code	13
How a Product Works with DevPay	14
Desktop Products	14
Web Products	16
Product Registration	17
Product Activation	18
Amazon S3 Requests	20
Where Do You Go From Here?	24
Please Provide Feedback	26
Amazon DevPay Resources	27
Document History	29
Document Conventions	30

Welcome

Topics

- [How Do I...? \(p. 1\)](#)

Amazon DevPay is an easy-to-use online billing and account management service that makes it easy for you to sell an Amazon Elastic Compute Cloud AMI or an application built on the Amazon Simple Storage Service.

This is the *Amazon DevPay Getting Started Guide*. This section describes who should read this guide, how the guide is organized, and other resources related to Amazon DevPay.

For a description of changes in this release of the *Amazon DevPay Getting Started Guide*, see [Document History \(p. 29\)](#).

How Do I...?

The typical getting started guide for AWS services shows how to use commands or create code in several programming languages to access the service. Because Amazon DevPay uses a web service but isn't a web service itself, this getting started guide is different. It instead discusses the general steps required to use DevPay. We recommend you use the guide to help you understand what is required at a high level and plan for your integration (for specific details, see the *Amazon DevPay Developer Guide*). After reading the guide, you should understand the major parts of your system that will be affected and the types of resources you need to move forward.

The guide is divided into the following major sections:

- [What Is Amazon DevPay? \(p. 3\)](#)
- [The Steps to Using DevPay \(p. 6\)](#)
- [Amazon S3 Example \(p. 12\)](#)
- [Where Do You Go From Here? \(p. 24\)](#)

"What is Amazon DevPay?" briefly describes DevPay and why you would use it.

"The Steps to Using DevPay" summarizes the overall process you follow to use DevPay. The specific details of the individual tasks are covered in the *Amazon DevPay Developer Guide*. We list the relevant sections in that guide that you should read when performing each task.

Amazon DevPay Getting Started Guide How Do I...?

"Amazon S3 Example" first gives an overview of how an Amazon S3 application works with DevPay. The section then discusses the specific changes necessary for an Amazon S3 application, using sample code in Java, C++, and Ruby. We don't include example code for an Amazon EC2 AMI because no coding is required to make an AMI work with DevPay (and the technical details are covered completely in the *Amazon DevPay Developer Guide*).

"Where You Go From Here" describes what we recommend you do next to get even more familiar with Amazon DevPay.

What Is Amazon DevPay?

Topics

- [Ways to Use DevPay \(p. 4\)](#)
- [Summary of Fees to Use DevPay \(p. 5\)](#)



Amazon DevPay is a billing and account management service that enables you to get paid for products you build on either Amazon EC2 or the Amazon Simple Storage Service. Amazon DevPay creates and manages the order pipeline and billing system for you. Your customers sign up for your product, and we automatically meter their usage of the underlying AWS service, bill them based on the pricing you set, and collect their payments. What else is special about DevPay?

- You can charge customers for your product; the charges can include recurring charges based on the customer's usage of AWS services (Amazon EC2 or Amazon S3), a fixed one-time charge, and a recurring monthly charge.
- Your customers can easily sign up and pay for your product with their trusted Amazon.com accounts; they do not need to sign up for an AWS developer account.
- Your customers are authenticated, thus ensuring they have access only to what they should.
- If your customers don't pay their bills, DevPay turns off their access to your product for you.
- Amazon Payments handles ALL payment processing.



Basic DevPay Flow

1	Your customer uses an Amazon.com account to sign up and pay for your product. The sign-up page indicates that you have teamed up with Amazon Payments to make billing easy and secure.
2	Your customer pays the price you've defined to use your product.

	DevPay subtracts a fixed transaction fee and pays you the difference.
	You pay the costs of the AWS services your product consumed, and a percentage-based DevPay fee.

Ways to Use DevPay

You can use DevPay with Amazon EC2 or Amazon S3.

With Amazon EC2

The most popular way to use Amazon EC2 and Amazon DevPay together is by selling an AMI you've created to other Amazon EC2 users.

An AMI that you sell through DevPay is called a *paid AMI*. Essentially you register a product with DevPay, receive a DevPay product code, and associate that product code with one or more of your AMIs (making them *paid AMIs*). Customers who buy your product launch instances of your paid AMIs (you don't launch instances on their behalf). When they do, they're charged the rates you've set, and not the normal Amazon EC2 rates.

The other way to use Amazon EC2 and Amazon DevPay together is with a *supported AMI*. With a supported AMI, you charge for software or a service you provide that other Amazon EC2 users use with their own AMIs. In this case, you register a product with DevPay, receive a DevPay product code, and the customers who buy your product associate the product code with *their own* AMIs. The AMIs must be Amazon S3-backed and not Amazon EBS-backed. When your customers launch instances of their AMIs, they're charged the rates you've set, and not the normal Amazon EC2 rates.

With the current implementation of Amazon DevPay:

- Your paid or supported AMIs must be backed by Amazon S3. Paid or supported AMIs backed by Amazon Elastic Block Store are currently not supported. Therefore, your paid AMIs cannot run Windows 2008 Server or SQL Server 2008 at this time.
- You can't use Elastic Load Balancing (either by itself or in conjunction with Auto Scaling) with instances of paid or supported AMIs.
- The discounts from Amazon EC2 Reserved Instances don't apply to paid or supported AMIs. That is, if you purchase Reserved Instances, you don't get the lower usage price associated with them when your customers launch your paid or supported AMIs. Also, if your customers purchase Reserved Instances, and they use your paid or supported AMIs, they continue to pay the price you specified for the use of your paid or supported AMIs.
- Your customers can't make Spot Instance requests for your paid or supported AMIs; if they do, Amazon EC2 returns an error.

For more information about any of the preceding Amazon EC2 features, go to the [Amazon EC2 product page](#).

For complete information about how paid AMIs and supported AMIs work, go to [Using DevPay with Your Amazon EC2 AMI](#) in the *Amazon DevPay Developer Guide*.

With Amazon S3

You can use Amazon DevPay with either a *desktop* product or a *hosted (web)* product that uses Amazon S3. To use your Amazon S3 product with DevPay:

- Your product must use the REST interface when calling Amazon S3 (web products may also use pre-signed URLs that customers use directly in a browser). SOAP is not supported.
- Your product must create a separate bucket in Amazon S3 for each customer who buys and uses the product. This is true even if your DevPay product is a web product. Each DevPay product can create up to 100 buckets per customer. For example, a customer who uses three different DevPay products can have up to 300 DevPay buckets, plus any other buckets created outside of DevPay (i.e., those created with a personal AWS account).

Once your product has created a bucket and put objects in it, only your product can access that bucket and the objects in it. For more information about restrictions on data access, go to the [section on data access](#) in the *Amazon DevPay Developer Guide*.

To use your product, your customers don't have to be signed up for Amazon S3. They don't need to obtain AWS credentials (like a Secret Access Key or Access Key ID); you obtain the credentials that DevPay requires and use them on your customers' behalf. Customers pay the rates you've set for your product, and not the normal Amazon S3 rates. It's your choice whether you let your customers know that the product uses Amazon S3.

When the customer uses the product, the product makes calls to Amazon S3 to store and manage objects on the customer's behalf. Therefore, the product must provide particular product and customer identifiers in the call so the correct customer can be billed. This guide covers what desktop and web products must do to meet those basic requirements (for more information, see [Amazon S3 Example \(p. 12\)](#)). For complete information about Amazon S3 DevPay products, go to [Using DevPay with Your Amazon S3 Product](#) in the *Amazon DevPay Developer Guide*.

Summary of Fees to Use DevPay

You pay:

- 3% of your *value-add* per customer
- \$0.30 per DevPay product for each customer bill we collect

Note

Your value-add per customer is the amount you charge each customer on top of the cost of the AWS services they used.

When we calculate the 3% fee amount, we use the value-add amount *before* any \$0.30 DevPay fees have been subtracted.

Complete details about the fees, how they're collected, and how you get paid are covered in the [Amazon DevPay Developer Guide](#).

The Steps to Using DevPay

Topics

- [Make Business Decisions about the Product \(p. 7\)](#)
- [Register Your Product with DevPay \(p. 7\)](#)
- [Prepare Your Amazon Payments Account \(p. 8\)](#)
- [Integrate Your Product with DevPay \(p. 8\)](#)
- [Update Your Product Support System \(Optional\) \(p. 9\)](#)
- [Update Your Web Site \(p. 9\)](#)
- [Test Your Entire System \(p. 10\)](#)
- [Make Any Changes to the Product \(p. 10\)](#)
- [Advertise and Sell Your Product to the Public \(p. 10\)](#)
- [Monitor Your Product's Usage and Get Paid \(p. 10\)](#)

This section summarizes the overall process you follow to use DevPay. The tasks are listed in the following table. For the specific details of the individual tasks, go to the [Amazon DevPay Developer Guide](#). We list the relevant sections in that guide that you should read when performing each task.

Note

The typical getting started guide for AWS services shows code for accessing the service. Because Amazon DevPay isn't a web service itself, this guide instead discusses the process for using DevPay. We recommend you use the guide to help you understand what is required at a high level and plan for your integration.

Overall Process to Use DevPay

1	Make business decisions about the product. (p. 7)
2	Register your product with DevPay. (p. 7)
3	Prepare your Amazon Payments account. (p. 8)
4	Integrate your product with DevPay. (p. 8)
5	Update your product support system (optional). (p. 9)
6	Update your web site. (p. 9)

7	Test your entire system. (p. 10)
8	Make any changes to the product. (p. 10)
9	Advertise and sell your product to the public. (p. 10)
10	Monitor your product's usage and get paid. (p. 10)

When actually performing these tasks, you'll need to follow the instructions in the *Amazon DevPay Developer Guide*.

Make Business Decisions about the Product

First, you must understand the business aspects of DevPay and make some decisions about the business side of your product. For example, you must determine the pricing for the product, if you want to provide customer support, the contact information for the product, etc.

Related Topics in the Amazon DevPay Developer Guide

- [The Business Aspects of DevPay](#)

Register Your Product with DevPay

Once you've determined the business information about the product, you register your product with DevPay.

Note

Registration *does not* make your product visible to the world. Customers can't sign up for it *until* you advertise the purchase URL (which is where customers sign up). AWS doesn't automatically advertise it anywhere. For more information about advertising, see [Advertise and Sell Your Product to the Public \(p. 10\)](#).

During registration, you provide the product's price, the product's description, any terms and conditions customers must agree to, contact information, and other product information. If your product uses Amazon EC2, you also specify which types of AMIs you want to sell and where (e.g., Linux/UNIX, Windows, in the U.S., Europe, etc.).

In return for registering your product, you receive three items:

- The *purchase URL* for the product, which is where customers sign up for the product
- A *product code*, which is an eight-character identifier for the product
- A *product token*, which is a long value prefixed with the literal string `{ProductToken}`

You need these items when updating your web site and integrating your product with DevPay.

Note

After you register your product, it must be approved by AWS. Product approval typically occurs within one business day. During this time, you can begin integrating your product with DevPay. When the product is approved, the purchase URL becomes functional. However, customers still can't purchase your product *until* you advertise the purchase URL yourself. AWS doesn't automatically advertise the URL anywhere.

Related Topics in the Amazon DevPay Developer Guide

- [Registering Your Product](#)

Prepare Your Amazon Payments Account

When you register your first DevPay product, we create an *Amazon Payments Business Account* for you. This account is associated with the login and password you used when you registered the product. In the documentation and elsewhere we refer to this account as your *Amazon Payments* account.

This task in the overall process of using DevPay covers a couple procedures that prepare your Amazon Payments account so it works with DevPay:

- Required: Verify your e-mail address with Amazon Payments
- Optional: Associate a bank account with your Amazon Payments account, and verify that bank account.

The instructions for these two tasks are covered as part of the product registration instructions (go to [Registering Your Product](#) in the *Amazon DevPay Developer Guide*). However, we call out the tasks separately here to emphasize their importance. If you don't verify your e-mail address with Amazon Payments, you can't use DevPay or Amazon Payments. If you don't associate a bank account with your Amazon Payments account and then verify that bank account, you can't withdraw money from your Amazon Payments account, and you can receive only up to \$10,000 total each month from all your DevPay customers.

Note

If you start to approach the \$10,000 limit during a given month, we'll send you e-mails that remind you to associate a bank account with your Amazon Payments account and then verify that bank account.

Related Topics in the Amazon DevPay Developer Guide

- [What Is Amazon Payments?](#)
- [Your Amazon Payments Account](#)
- [Registering Your Product](#)

Integrate Your Product with DevPay

After you register your product, you can begin integrating the product with DevPay. The task varies based on whether your product uses Amazon EC2 or the Amazon Simple Storage Service.

Amazon EC2

If your product uses Amazon EC2, you simply associate your product code with each AMI that you want to sell. Amazon EC2 provides a command line function and an API function for this.

Related Topics in the Amazon DevPay Developer Guide

- [Using DevPay with Your Amazon EC2 AMI](#)

Amazon S3

If your product is a desktop or web product using Amazon S3, you complete the tasks in the following table.

Process to Integrate an Amazon S3 Product

1	You embed the product token in the code of the product so the token is easily retrievable when the product makes any AWS requests.
2	You update your application to call the License Service each time a customer signs up for the product (we provide a library for the License Service in the sample code). The License Service activates your product for use by that particular customer. <i>Activation</i> means you retrieve credentials required to make Amazon S3 requests on behalf of that customer.
3	You update your own Amazon S3 library or application to make Amazon S3 requests as required by DevPay. This involves: <ul style="list-style-type: none">• Adding two new HTTP headers to the request• Adding the two new headers to the string that is the basis of the request's HMAC signature We've provided a simple Amazon S3 library that includes these changes.

Related Topics in the Amazon DevPay Developer Guide

- [Using DevPay with Your Amazon S3 Product](#)

Update Your Product Support System (Optional)

If you want to restrict customer support just to those who are currently subscribed to your product, you can add a programmatic function that verifies your customers' status. Simply update your product support system to use the License Service. The service provides a function that returns the current subscription status of a customer.

Related Topics in the Amazon DevPay Developer Guide

- [Verifying the Customer's Subscription Status](#)

Update Your Web Site

You need to update your web site to include information about your product, the purchase URL, contact information, and where customers can get information about their bills for your product.

Related Topics in the Amazon DevPay Developer Guide

- [Customer Support](#)
- [Recommendations for Product Development](#)

Test Your Entire System

When testing, include your web site, your purchase pipeline, your product's integration with DevPay and the License Service, and your customer support system. Amazon DevPay does not provide a sandbox; therefore your testing is on your own live product (recall that the public doesn't know where to purchase your product until you tell them).

Related Topics in the Amazon DevPay Developer Guide

- [Testing and Going Live](#)

Make Any Changes to the Product

You might find during testing that you want to change the price or other information about the product. Make those changes now.

Related Topics in the Amazon DevPay Developer Guide

- [Making Changes to Your Product](#)
- [Changing Pricing](#)

Advertise and Sell Your Product to the Public

When you're ready to go into production, you make the purchase URL available to the public. AWS doesn't automatically advertise your product or purchase URL anywhere. It's up to you to do it.

To advertise an Amazon S3 application

- Post information about the application in the [Solutions Catalog](#) on the AWS Developer Connection site.

To advertise an Amazon EC2 paid AMI

- Post information about the AMI [Solutions Catalog](#) on the AWS Developer Connection site and on the [Amazon Machine Images \(AMIs\)](#) page on the AWS Resource Center.

For information about co-marketing your product with AWS, go to the [co-marketing area of the Developer Connection site](#).

Monitor Your Product's Usage and Get Paid

When customers start purchasing and using your product, you can view usage data and the corresponding revenue you expect to receive. This information is displayed on your DevPay Activity page (at <http://aws.amazon.com/devpayactivity>). When the monthly billing cycle occurs, your customers are billed for their usage and you receive their payments (minus the fixed DevPay transaction fee). Also, you're charged for the costs of the AWS service your product has used and the percentage-based DevPay fee. You can view your DevPay transaction history and your Amazon Payments account balance. You can also monitor how much of your expected revenue for a given month we've actually collected.

Related Topics in the Amazon DevPay Developer Guide

- [Your Amazon Payments Account](#)
- [When and How You Get Paid](#)
- [Appendix: Example DevPay Activity Pages](#)
- [Reports](#)

Amazon S3 Example

Topics

- [About the Sample Code](#) (p. 13)
- [How a Product Works with DevPay](#) (p. 14)
- [Product Registration](#) (p. 17)
- [Product Activation](#) (p. 18)
- [Amazon S3 Requests](#) (p. 20)

This section describes how to modify an existing Amazon Simple Storage Service example library so that it works with Amazon DevPay. It is assumed that you understand how the integration work outlined in this section fits in with the overall process of using DevPay (for more information, see [The Steps to Using DevPay](#) (p. 6)).

Note

Integrating an Amazon S3 product with DevPay requires coding, whereas integrating an Amazon EC2 AMI with DevPay does not. Therefore, in this section we only explain how to make an *Amazon S3* product work with DevPay.

"About the Sample Code" gives information about the sample code, such as its location, how we created it, and the prerequisites for using it.

"How a Product Works with DevPay" describes in general how Amazon DevPay works with desktop products and web products. You need to understand this overall process before focusing on the specific code changes for your own product.

"Product Registration" briefly describes product registration and how it's relevant to the sample code.

"Product Activation" describes the initial process your product goes through with each new customer who signs up. Product activation involves making an API call to the License Service. Snippets of sample code are presented showing the API call.

"Amazon S3 Requests" describes the changes you must make to an existing Amazon Simple Storage Service library to make calls as required by DevPay. Snippets of sample code are presented showing the required changes.

About the Sample Code

The sample code is available in the [Amazon DevPay Community Code](#) area of the AWS Resource Center. A separate download is posted for each available language (Java, C#, C++, and Ruby).

To create the sample code for all languages except C++, we started with an existing Amazon Simple Storage Service library (to get the library, go to the [Amazon Simple Storage Service Community Code](#) area of the AWS Resource Center and look for the one called "Amazon S3 Library for REST in [language]"). We then updated that library so that it makes Amazon S3 requests as required by DevPay (the updated library is in the `s3` directory of the DevPay sample code package for each language). We also added an additional library (in the `ls` directory of the sample code) that calls the License Service. There was no existing Amazon S3 C++ library for us to start with, so we simply created a new one that works with DevPay.

For the Java, C#, and Ruby sample code, you can search for the differences between the original Amazon S3 library and the new DevPay version of the library to easily see what we've changed. The changes are in the `AWSAuthConnection` class.

In this guide, the Java, C#, and C++ samples demonstrate what is required to make a desktop product (or any distributed product that resides on a desktop, handheld device, etc.) work with DevPay. The Ruby sample demonstrates what is required to make a web (hosted) product work with DevPay. For information about how to handle desktop products and web products, see [How a Product Works with DevPay](#) (p. 14).

Prerequisites for the Sample Code

The following sections list the prerequisites to use the sample code discussed in this guide.

Java (Desktop Product)

- You must have Apache Ant (for more information, go to <http://ant.apache.org>).
- You must be signed up to use Amazon S3.

Note

The Java sample has been tested with Java 6 only.

C# (Desktop Product)

- You must have Microsoft Visual Studio 2005 (for more information, go to <http://msdn2.microsoft.com/en-us/vs2005/default.aspx>).
- You must be signed up to use Amazon S3.

C++ (Desktop Product)

- You must have the following libraries. They are typically included with Linux.
 - libcurl (for more information, go to <http://curl.haxx.se/libcurl>)
 - expat (for more information, go to <http://expat.sourceforge.net>)
- You must be signed up to use Amazon S3.

Ruby (Web Product)

- You must be signed up to use Amazon S3.

About the Shell Program

The sample code includes a simple command shell program that uses the libraries. The shell is located in the `ui` directory in the sample code package.

Important

To use the shell, you must sign up for a sample DevPay product (like your customers would sign up for your DevPay product). This sample DevPay product has a \$0.35 sign-up charge and a monthly recurring charge of \$0.35. When you sign up, you are charged accordingly. If you do not want to sign up for the sample product and use the shell, you can still look at all the sample code provided in the package. If you'd like to use the shell for a limited time, you can sign up and then cancel your use of the sample DevPay product when you're done with it.

For details about how to use the shell, see the README file in the sample code package.

How a Product Works with DevPay

Topics

- [Desktop Products \(p. 14\)](#)
- [Web Products \(p. 16\)](#)

How your Amazon Simple Storage Service product works with Amazon DevPay varies based on whether the product is a *desktop product* (or any distributed product that resides on a desktop, handheld device, etc.) or a *web (hosted) product*.

Desktop Products

This section describes how desktop products meet the DevPay requirements to ensure the correct customer is billed for use of the product. For additional requirements applicable to Amazon S3 products that use DevPay, see [Ways to Use DevPay \(p. 4\)](#).

Following are the main differences between an Amazon DevPay desktop product and a regular desktop product:

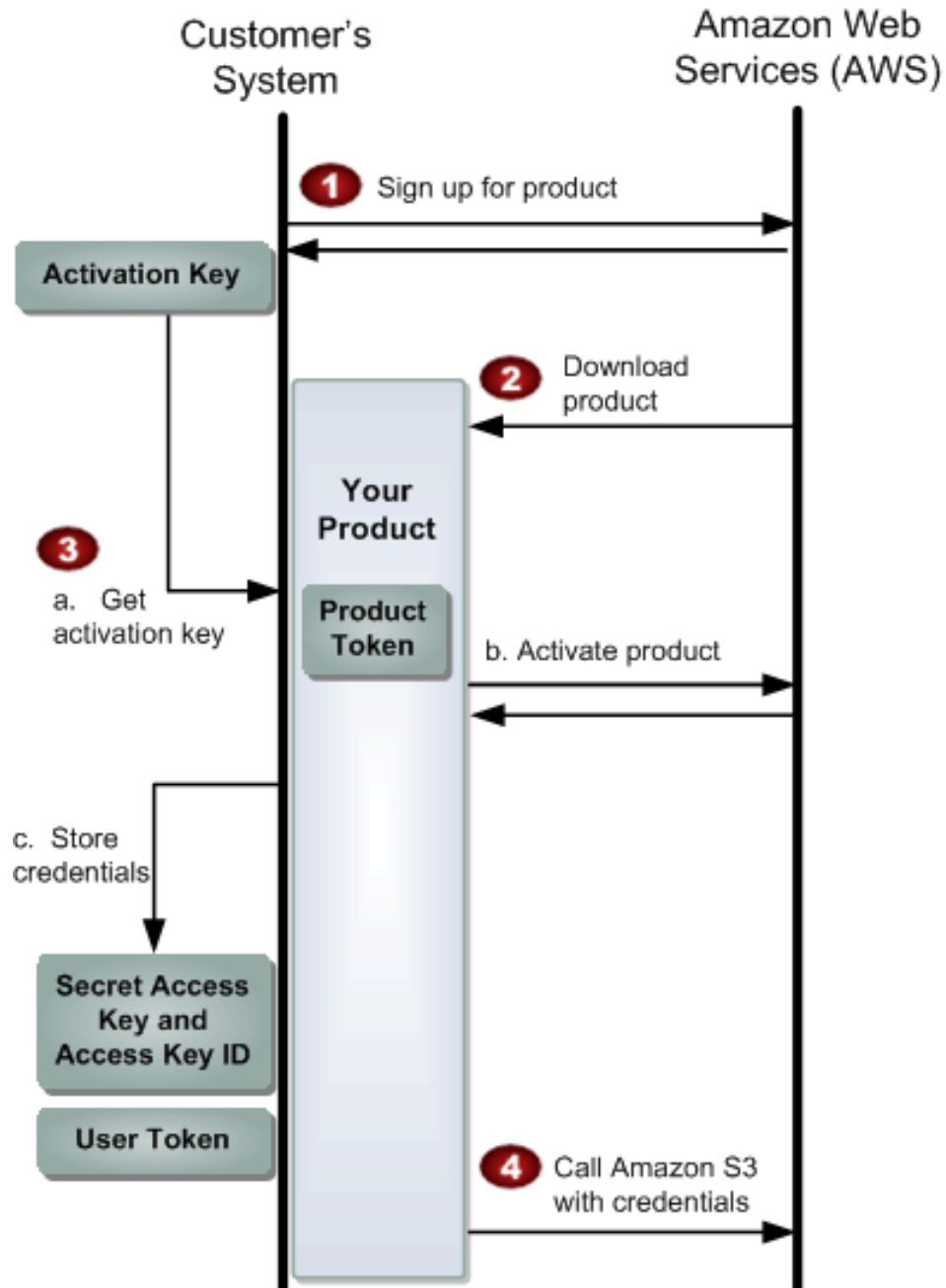
- The DevPay desktop product must have the DevPay *product token* embedded in it so it can provide the product token each time it makes a request to AWS (you obtain the product token when you register the product with DevPay; for more information, see [Product Registration \(p. 17\)](#)). The product token is a long encoded string prefixed with the literal string `{ProductToken}`.
- The DevPay desktop product must call the License Service to activate itself and obtain a set of credentials for the specific customer using the product. The product must store those credentials securely on the customer's system and use them when making a request to Amazon S3 on behalf of that customer. The credentials include a *user token* (a long encoded string prefixed with the literal string `{UserToken}`), a Secret Access Key, and an Access Key ID. The product includes the user token in the Amazon S3 request and signs the request using the customer's Secret Access Key (instead of your own).
- The DevPay desktop product must include two additional headers in the REST Amazon S3 request (one for the product token and one for the user token).

Important

The aforementioned customer credentials are specific to your product. If John Smith signs up to use both your DevPay product and another DevPay product, he will have a similar but separate set of credentials for each product stored on his system. Each DevPay product must be able to

keep its customer credentials separate from those of another DevPay product installed on the customer's system.

The preceding items are required to make your product handle customer authentication correctly for DevPay. The overall process of customer authentication for a desktop product is described in the diagram and corresponding steps that follow.



Overall Authentication Process for Desktop Products

1	The customer signs up for the product by clicking the purchase URL AWS provided you during product registration, and as part of the process, the customer: <ul style="list-style-type: none">• Logs in with an Amazon.com login (or creates a new login; this a regular Amazon.com login and <i>not</i> an AWS developer account)• Signs up to use your product• Obtains an <i>activation key</i> required by your product
2	The customer downloads and installs your product.
3	As part of the installation, or immediately after, the product goes through an activation process: <ol style="list-style-type: none">a. Your product obtains the activation key from the customer.b. Your product sends a request to the License Service to activate itself and obtain a Secret Access Key, Access Key ID, and user token for the customer. Your request includes the product token for your product and the customer's activation key.c. Your product appropriately stores the credentials it has received so the customer can use the product seamlessly in the future.
4	Later, when the customer uses the product, the product makes an Amazon S3 request on behalf of the customer. In the process, the product retrieves and uses the Secret Access Key, Access Key ID, user token, and the product token for the product.

Web Products

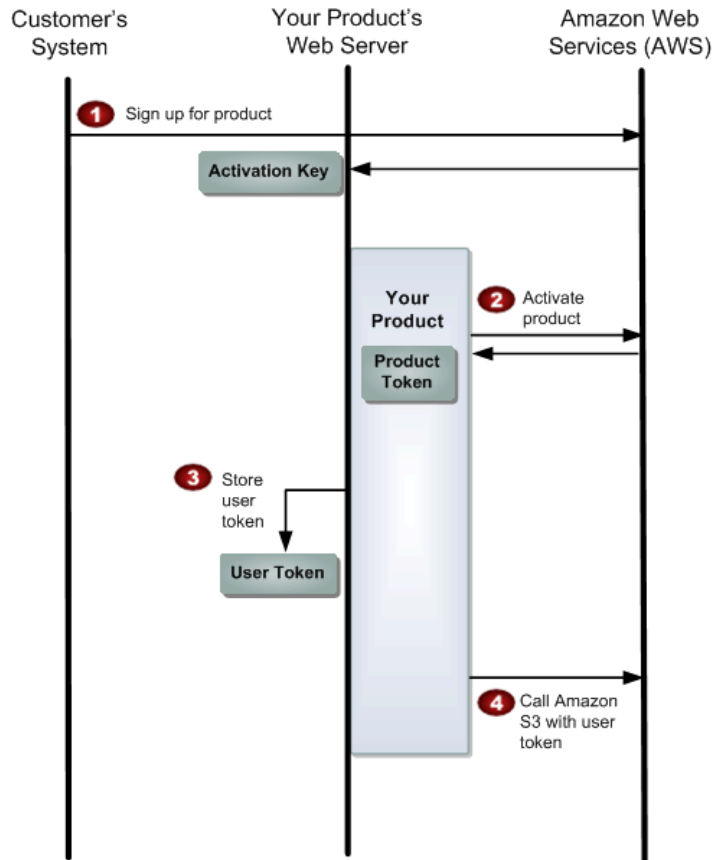
This section describes how web products meet the DevPay requirements to ensure the correct customer is billed for use of the product. For additional requirements applicable to Amazon S3 products that use DevPay, see [Ways to Use DevPay \(p. 4\)](#).

Following are the main differences between an Amazon DevPay web product and a regular web product:

- The DevPay web product must have the DevPay *product token* embedded in it so it can use the product token as required in requests to AWS (you obtain the product token when you register the product with DevPay; for more information, see [Product Registration \(p. 17\)](#)).
- The DevPay web product must call the License Service to activate itself and obtain a *user token* for the specific customer using the product. The product must store the user token securely and include it in each request to the Amazon Simple Storage Service on behalf of that customer.
- The DevPay web product must include an additional header in the REST Amazon S3 request (to pass the user token and optional product token).

The preceding items are required to make your product handle customer authentication correctly for DevPay. The overall process of customer authentication for a web product is described in the diagram and corresponding steps that follow.

Amazon DevPay Getting Started Guide Product Registration



Overall Process of Authentication for Web Products

1	The customer signs up for the product by clicking the purchase URL AWS provided you during product registration. When the customer completes the purchase, AWS generates an activation key for that customer and makes it available to your server.
2	Your product sends a signed request to the License Service to activate itself and obtain a user token for the customer. The request includes the product token for your product and the activation key.
3	Your product appropriately stores the user token it has received. Your product should associate the user token with the customer who is logged in to your web product.
4	Later, when the customer uses the product, the product makes an Amazon S3 request on behalf of the customer. In the process, the product retrieves and includes the customer's user token and the product token in the request. Note The product token is optional in the REST request if the web product has a user token created after May 15, 2008.

Product Registration

If you want a product you've created to work with Amazon DevPay, you must register it with DevPay at some point before you go live with the product. Registration *does not* make your product visible to the

world (you do that yourself when you advertise the URL where customers can sign up for your product). For complete information about registration, go to the [Amazon DevPay Developer Guide](#).

For this example, all you need to know is that the registration process provides you with:

- **Purchase URL**— A URL for a web page where customers can purchase your product
- **Product token**— A token that identifies your product to AWS. It is a long encoded string prefixed with the literal string `{ProductToken}`.
You must embed the product token within the code of the product because the product token is used in subsequent requests to AWS.

For the purposes of this example, we'll proceed as if the registration process for the sample code is complete and the product token is already embedded into the sample code.

Note

During product registration, you're also provided with a *product code*, but it's not needed for this particular example.

Product Activation

Your product must go through a process of *activation* before each customer can use it. This process is part of the overall process products follow to work with Amazon DevPay (for more information, see [How a Product Works with DevPay \(p. 14\)](#)). This section describes how the product activates itself and obtains the required credential or credentials for the customer.

Prompt the Customer for the Activation Key

When the customer launches the product the first time, the product must get the *activation key* that Amazon provided to the customer during sign-up. The sample code presented here assumes that the activation key has already been obtained. For information about the different ways you can obtain the activation key, go to [Desktop Product Activation](#) or [Web Product Activation](#) in the *Amazon DevPay Developer Guide*.

Obtain the Credentials (Activate the Product)

The product now needs to make a *REST-Query* call to the License Service to obtain the required credential (for a web product) or credentials (for a desktop product). This is also known as *activating* the product. The request includes these items:

- The customer's activation key
- The *product token*

The response contains these items:

- The customer's Secret Access Key and Access Key ID (for desktop products only)
- The customer's *user token*

Your product must encode and securely store these items. For an example of one way to do it, see the `FileCredentialStore` class in the `security` directory of the sample code package.

Java (Desktop Product)

```
// Note that the generated License Service library expects to sign all
// requests to AWS services. Because ActivateDesktopProduct is an unsigned
// call, we pass dummy signing credentials.

AmazonLSConfig config = new AmazonLSConfig();
    config.setServiceURL("https://ls.amazonaws.com");

ActivateDesktopProduct action = new ActivateDesktopProduct();

AmazonLS service = new AmazonLSQuery("dummy", "dummy", config);
ActivateDesktopProductResponse response =
service.activateDesktopProduct(action.withActivationKey(activationKey_).withPro
ductToken(productToken_));

if (response.isSetActivateDesktopProductResult()) {
    ActivateDesktopProductResult activateDesktopProductResult =
        response.getActivateDesktopProductResult();

    // if product is activated, get credentials into private variables
    if (activateDesktopProductResult.isSetUserToken()) {
        userToken_ = activateDesktopProductResult.getUserToken();
    }
    if (activateDesktopProductResult.isSetAWSAccessKeyId()) {
        awsAccessKeyId_ =
            activateDesktopProductResult.getAWSAccessKeyId();
    }
    if (activateDesktopProductResult.isSetSecretAccessKey()) {
        awsSecretAccessKey_ =
            activateDesktopProductResult.getSecretAccessKey();
    }
}
```

C# (Desktop Product)

```
// Note that the generated License Service library expects to sign all
// requests to AWS services. Because ActivateDesktopProduct is an unsigned
// call, we pass dummy signing credentials.

AmazonLS service = new AmazonLSQuery("dummy", "dummy");

ActivateDesktopProduct action = new ActivateDesktopProduct();
action.ActivationKey = activationKey;
action.ProductToken = productToken;

ActivateDesktopProductResponse response = service.ActivateDesktopProduct(action);
return new Credentials(
    response.ActivateDesktopProductResult.UserToken,
    response.ActivateDesktopProductResult.AWSAccessKeyId,
    response.ActivateDesktopProductResult.SecretAccessKey
);
```

C++ (Desktop Product)

```
AWSQueryConnection conn;  
ActivateResponse *response =  
    conn.activateDesktopProduct(activationKey, PRODUCT_TOKEN);  
userToken_ = response->userToken;  
awsAccessKeyId_ = response->awsAccessKeyId;  
awsSecretAccessKey_ = response->awsSecretAccessKey;
```

Ruby (Web Product)

```
user_token = LS::Service.activate_hosted_product(activation_key,  
product_token,  
access_key_id,  
secret_access_key)
```

Important

You must design your web product to:

- Recognize when Customer A logs in to your site
- Use Customer A's user token (and not another customer's) when making Amazon S3 requests for Customer A

Amazon S3 Requests

Your product will make calls to the Amazon Simple Storage Service on behalf of a customer. These calls are part of the overall process products follow to work with Amazon DevPay (for more information, see [How a Product Works with DevPay \(p. 14\)](#)).

This section describes how the product makes requests to Amazon S3 and uses the customer's credentials. For the Java, C#, and Ruby sample code, you can search for the differences between the original Amazon S3 library and the new DevPay version of the library to easily see what we've changed. The changes are in the `AWSAuthConnection` class. For information about where to get the original Amazon S3 library, see [About the Sample Code \(p. 13\)](#).

Retrieve the Credentials

For desktop products, the credentials your product must retrieve are the customer's Access Key ID, Secret Access Key, and user token. For web products, the only credential is the customer's user token. Your product must retrieve and decode the credentials from their secure store location. How you store and retrieve them is up to you. For an example of one way to handle them, see the `FileCredentialStore` class in the `security` directory in the sample code package.

Add the Tokens to the Amazon S3 Request

Desktop products and web products must include the customer's user token in the Amazon S3 request. In addition, desktop products must include the product token (it's optional for web products that have a user token created after May 15, 2008). To include the tokens in the request, you add an `x-amz-security-token` header for each token. The following example shows a basic REST request (with the additional headers in red).


```
Date: Wed, 27 Jun 2007 03:40:41 GMT
Authorization: AWS OPN6J17HBGXHT7JJ3X83:frJIUN8DYpKDtOLCwo//y1lqDzgEXAMPLE=
x-amz-security-token: {UserToken}AAAHVXNlclRrbgfOpSykBAXO7g/zG....[long encoded
token]...
x-amz-security-token: {ProductToken}MIIBzTCCATagAwIBAgIGARB1qe....[long en
coded token]...
```

An alternate method for passing both tokens is to add a single `x-amz-security-token` header with the tokens separated by a comma.

The following shows how to add the tokens to the request.

Java (Desktop Product)

For Java, make sure you *add* the two `x-amz-security-token` headers and not *put* them. Because the headers have the same name, if you put them, you'll overwrite the first with the second.

The following code snippet uses a single `x-amz-security-token` header with the values separated by a comma.

```
URLConnection connection = ...;
connection.setRequestProperty("x-amz-security-token", userToken + "," + product
Token);
```

C# (Desktop Product)

```
HttpWebRequest req = (HttpWebRequest)WebRequest.Create(url);
req.Headers.Add("x-amz-security-token", userToken + "," + productToken);
```

C++ (Desktop Product)

The following code snippet shows the method in the C++ sample code that performs the initial step to get the headers added to the request (this code creates a map; see `CppDevPaySample\s3\AWSAuthConnection.cpp`). The line in red near the end of the snippet adds the user token and product token.

```
void AWSAuthConnection::addHeaders(REQUEST_TYPE method, int length, strmap
&headers)
{
    string temp;
    //Add standard headers to the existing ones
    headers.insert(pair<string, string> (string("Host"), server_));
    temp.erase(); temp = utils_.getHTTPDate();
    headers.insert(pair<string, string> (string("Date"), temp));

    switch(method)
    {
        case PUT:
            if(length > 0)
            {
                headers.insert(pair<string, string> (string("Content-Type"),string("bin
ary"))));
            }
    }
```

```
        else
        {
            headers.insert(pair<string, string> (string("Content-Type"), string("")));
        }
        headers.insert(pair<string, string> (string("Content-length"),
utils_.itos(length)));
        headers.insert(pair<string, string> (string("Content-md5"), string("")));

        break;
    case GET:
    case DELETE:
        headers.insert(pair<string, string> (string("Content-Type"), string("")));

        headers.insert(pair<string, string> (string("Content-md5"), string("")));

        break;
    }
    if(!securityTokens_.empty())
    {
        set<string>::iterator i;
        for(i = securityTokens_.begin(); i != securityTokens_.end(); i++)
        {
            headers.insert(pair<string, string> (utils_.AMAZON_SECURITY_HEADER, (*i)));
        }
    }
    return;
```

Ruby (Web Product)

The following code snippet uses a single `x-amz-security-token` header with the values separated by a comma.

```
http = Net::HTTP.new(server, port)
request = Net::HTTP::Get.new(path)
...
request['x-amz-security-token'] = product_token + "," + user_token
...
http.request(request)
```

Sign the Request

How you sign the Amazon S3 request for a DevPay product is essentially no different from how you do it for a product that doesn't use DevPay. You still sign the request with a Secret Access Key and include an Access Key ID in the `Authorization` header. For desktop products, you use the customer's Secret Access Key and Access Key ID. For web products, you use your Secret Access Key and Access Key ID.

When creating the string to sign, you still include any headers that start with `x-amz`, as discussed in the Amazon S3 documentation. The two new headers you added in the previous section start with `x-amz`, so they are included when the string to sign is created by your code. For information about how to sign an Amazon S3 request, go to the [Amazon Simple Storage Service Developer Guide](#). Also refer to the basic Amazon S3 libraries (for information about their location, see [About the Sample Code \(p. 13\)](#)).

Java (Desktop Product)

The existing Java Amazon S3 library that we started with did not have to be modified because all the headers that begin with `x-amz` were already included in the string.

C# (Desktop Product)

The existing C# Amazon S3 library that we started with did not have to be modified because all the headers that begin with `x-amz` were already included in the string.

C++ (Desktop Product)

The C++ sample code handles the `x-amz-security-token` headers separately from the other headers that begin with `x-amz`. The following snippet is from `CppDevPaySample\s3\Utils.cpp`.

```
1AMAZON_SECURITY_HEADER = "x-amz-security-token";
2...
3if(temp.compare(0, AMAZON_SECURITY_HEADER.size(), AMAZON_SECURITY_HEADER) ==
40)
5{
6    if(isSecurityHeaderAdded == false)
7    {
8        bool isFirst = true;
9        ret = sortedheaders.equal_range(AMAZON_SECURITY_HEADER);
10       result.append(AMAZON_SECURITY_HEADER);
11       for(j = ret.first; j != ret.second; j++)
12       {
13           result.append((isFirst == true) ? ":" : ",");
14           trimmedval.erase();
15           trimmedval = trimString((*j).second);
16           result.append(trimmedval);
17           if(isFirst == true)
18               isFirst = false;
19       }
20       result.append("\n");
21       isSecurityHeaderAdded = true;w
22   }
23 LASTLINE
```

Ruby (Web Product)

The existing Ruby Amazon S3 library that we started with did not have to be modified because all the headers that begin with `x-amz` were already included in the string.

Where Do You Go From Here?

Now that you've read through this guide, you have a good idea of the main tasks you need to perform to use Amazon DevPay, and where to go in the *Amazon DevPay Developer Guide* for more information and instructions. This section describes the next steps we recommend you take.

Register a Product

If you want to jump right in, you can go ahead and register a product with DevPay. This will give you direct experience with pricing a product. You can change any aspect of your product (the name, price, etc.) after it's registered, so you don't need to have all that final information now. For instructions on registering a product, go to "[Registering Your Product](#)" in the *Amazon DevPay Developer Guide*.

After you complete the registration, your purchase URL is available. The aforementioned "Registering Your Product" topic tells you how to get your purchase URL. The URL is the link customers click to purchase your product. Note that the purchase URL *is not live yet*, and it hasn't been published anywhere. The URL doesn't become functional until AWS reviews and approves your product (a process that takes a few business days; you get an e-mail when the product is approved). Even after the URL becomes functional, only you know about the URL. AWS does not list it anywhere publicly. So only you can sign up for your product at this point. When you're ready to sell the product, then you (and not AWS) advertise the URL.

We recommend you use the purchase URL and buy your own product. This lets you see what the purchase experience is like for your customers.

Sign Up for Demo Products

If you want to see other products' purchase pipelines, there are two demonstration DevPay products you can purchase. One is for an Amazon EC2 demo paid AMI, and the other is for an Amazon S3 demo application.

For information about the demo paid AMI, go to [the article "Using Paid AMIs and the Demo Paid AMI"](#) on the AWS web site. It includes a link to a demo paid AMI. The demo paid AMI costs \$0.11 per instance-hour consumed, \$0.11 per GB data transfer in, and \$0.20 per GB data transfer out. There is no monthly charge or one-time charge for it.

For information about the demo Amazon S3 application, see [About the Sample Code \(p. 13\)](#). You can sign up for the demo application without actually downloading and using it. To only sign up for it, click its [purchase URL](#). The product has a \$0.35 sign-up charge and a monthly recurring charge of \$0.35. When you sign up, you are charged accordingly. If you want to actually *get and use* the demo application, you must download one of the DevPay sample code packages. The package includes the application (called "the shell program") and a ReadMe file that has instructions for using the program. If you use it, you're charged according to the prices shown when you signed up for the product.

When you sign up for one of these demos, you're signing up for a real DevPay product that costs money (albeit a small amount). If you use the demo paid AMI or application after signing up for it, your credit card will be charged accordingly for the usage. You'll get a separate monthly bill that covers all the DevPay products you've bought (including your own DevPay products, if you've bought any). You can view information about the products by going to your Application Billing page at <http://www.amazon.com/dp-applications>. Each of your customers receives their own application billing page like this where they manage all the DevPay products they've bought.

We recommend you use the following procedure to cancel your subscription to the demo after you're done trying it out.

Important

If you store any data in Amazon S3 when using the demo Amazon S3 application, you won't be able to access that data in Amazon S3 after you cancel your subscription to the demo product. For information about why, go to [Customer Access to Data Stored by Your Amazon S3 Product](#) in the *Amazon DevPay Developer Guide*.

To cancel a subscription to a DevPay product

1. Go to your Application Billing page at <http://www.amazon.com/dp-applications>.
2. Log in with the Amazon.com login and password you used when you signed up for the demo product.
3. Click **View/Cancel Application** for the product you want to cancel.
You are then prompted to confirm the cancellation.
4. Click **Cancel this Application**.

If the demo product has a monthly fee, AWS refunds you the unused portion of that monthly fee.

Read the Forum

We recommend you look at the [DevPay forum](#) to get an idea of what other users are doing and questions they've had. This will help you further understand what you can and can't do with Amazon DevPay.

Look at Other Available Sample Code

If you're building a DevPay product that uses Amazon S3, you're already aware of the sample code that goes with this guide (for more information, see [About the Sample Code \(p. 13\)](#)). You can look at any other sample code that's available from the [Sample Code and Libraries](#) page.

Please Provide Feedback

Your input is important to help make our documentation helpful and easy to use. Please tell us about your experience getting started with DevPay by completing our [Getting Started Survey](#).

Thank you.

Amazon DevPay Resources

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

Resource	Description
Amazon DevPay Developer Guide	The developer guide provides a detailed discussion of the all aspects of the service. It includes important information about the business aspects of DevPay, and all the technical details, including an API reference.
Amazon DevPay Release Notes	The Release Notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
Technical documentation for related AWS services	You might need the documentation for Amazon EC2 or Amazon S3. For those documents, go to the http://aws.amazon.com/documentation .
AWS Developer Resource Center	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
AWS Support Center	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and AWS Premium Support (if you are subscribed to this program).
Product information about Amazon DevPay	The primary web page for information about Amazon DevPay.
E-mail address for questions related to your DevPay product: <DevPay@amazon.com>	This e-mail address is <i>only</i> for questions about the business side of your DevPay product. For technical questions, use the Discussion Forums.
Contact Us	A central contact point for inquiries concerning AWS billing, account, events, abuse, etc.

Resource	Description
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com.

Document History

This documentation is associated with the 2007-12-01 release of Amazon DevPay. This guide was last updated on 16 May 2013.

Change	Description	Release Date
General Update	Added an expanded breadcrumb to the HTML version, and reduced the front matter content.	in this release
Amazon EC2 Spot Instances	Updated the information about paid and supported AMIs to reflect that DevPay does not support Spot Instances. For more information, see Ways to Use DevPay (p. 4) .	13 December 2009
Amazon EBS-Backed AMIs	Updated the information about paid and supported AMIs to reflect that DevPay only supports Amazon S3-backed AMIs (those that use an instance store root device), and not Amazon EBS-backed AMIs (those that use an Amazon EBS root device).	2 December 2009

Document Conventions

This section lists the abbreviations, typographical and symbol use conventions used in this guide.

Typographical Conventions

This section describes common typographical use conventions.

The following Amazon Web Services (AWS) products will occasionally be referred to using the following abbreviated forms; all copyrights and legal protections still apply.

Convention	Description/Example
Call-outs	<p>A call-out is a number in the body text to give you a visual reference. The reference point is for further discussion elsewhere.</p> <p>You can use this resource regularly. 1</p>
Code in text	<p>Inline code samples (including XML) and commands are identified with a special font.</p> <p>You can use the command <code>java -version</code>.</p>
Code blocks	<p>Blocks of sample code are set apart from the body and marked accordingly.</p> <pre># ls -l /var/www/html/index.html -rw-rw-r-- 1 root root 1872 Jun 21 09:33 /var/www/html/index.html # date Wed Jun 21 09:33:42 EDT 2006</pre>
Emphasis	<p>Unusual or important words and phrases are marked with a special font.</p> <p>You <i>must</i> sign up for an account before you can use the service.</p>
Internal cross references	<p>References to a section in the same document are marked.</p> <p>See Document Conventions (p. 30).</p>

Amazon DevPay Getting Started Guide Typographical Conventions

Convention	Description/Example
Logical values, constants, and regular expressions, abstracta	A special font is used for expressions that are important to identify, but are not code. If the value is <code>null</code> , the returned response will be <code>false</code> .
Product and feature names	Named AWS products and features are identified on first use. Create an Amazon Machine Image (AMI).
Operations	In-text references to operations. Use the <code>GetHITResponse</code> operation.
Parameters	In-text references to parameters. The operation accepts the parameter <code>AccountID</code> .
Response elements	In-text references to responses. A container for one <code>CollectionParent</code> and one or more <code>CollectionItems</code> .
Technical publication references	References to other AWS publications. If the reference is hyperlinked, it is also underscored. For detailed conceptual information, see the <i>Amazon Mechanical Turk Developer Guide</i> .
User entered values	A special font marks text that the user types. At the password prompt, type <code>MyPassword</code> .
User interface controls and labels	Denotes named items on the UI for easy identification. On the File menu, click Properties .
Variables	When you see this style, you must change the value of the content when you copy the text of a sample to a command line. <code>% ec2-register <your-s3-bucket>/image.manifest</code> See also the following symbol convention.

Symbol Conventions

This section describes the common use of symbols.

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

Document Abbreviations

Full Name	Abbreviated Form
Amazon DevPay	DevPay
Amazon Elastic Compute Cloud	Amazon EC2
Amazon Simple Storage Service	Amazon S3
Amazon Machine Image	AMI

The following Amazon Web Services (AWS) products will occasionally be referred to using the following abbreviated forms; all copyrights and legal protections still apply.

Full Name	Abbreviated Form
Amazon DevPay	DevPay
Amazon Elastic Compute Cloud	Amazon EC2
Amazon Simple Storage Service	Amazon S3

Amazon DevPay Getting Started Guide
Symbol Conventions

Full Name	Abbreviated Form
Amazon Machine Image	AMI