

---

# **Amazon Simple Queue Service**

## **Getting Started Guide**

**API Version 2011-10-01**



# **Amazon Simple Queue Service: Getting Started Guide**

Copyright © 2011 Amazon Web Services LLC or its affiliates. All rights reserved.

## Table of Contents

Welcome .....	1
Introduction to Amazon SQS .....	3
Getting Set Up .....	7
Creating an AWS Account .....	7
Signing Up for SQS .....	8
Getting Your AWS Identifiers .....	9
Getting the Tools You Need .....	11
Working with Amazon SQS .....	13
Preparing the Samples .....	13
Creating a Queue .....	18
Confirming the Queue Exists .....	21
Sending a Message .....	24
Receiving a Message .....	29
Deleting a Message .....	35
You're Finished! .....	39
Please Give Us Your Feedback .....	39
Where Do I Go from Here? .....	40
Document History .....	42
Glossary .....	43

---

# Welcome

---

## Topics

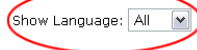
- [Showing Your Preferred Programming Language \(p. 1\)](#)
- [Amazon SQS Resources \(p. 2\)](#)

Welcome to the *Amazon Simple Queue Service* Getting Started Guide. Amazon Simple Queue Service is a messaging queue service: it's a service that handles message or work flows between other components in a system.

The following video walks you through the example presented in this guide: [Getting Started with Amazon SQS](#).

## Showing Your Preferred Programming Language

In the HTML version of this document, you can hide the sections of this guide that don't apply to the programming language you are using. There is a language selection menu in the upper-right corner of pages with language-specific text. Select your language to hide all others, or select All to show the examples in all available languages.



Your selection applies to all of the pages in this guide.

## Amazon SQS Resources

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

Resource	Description
<a href="#">Amazon Simple Queue Service Developer Guide</a>	The developer guide provides a detailed discussion of the service. It includes an architectural overview, programming reference, and API reference.
<a href="#">Amazon Simple Queue Service API Reference</a>	The API reference gives the WSDL location; complete descriptions of the API actions, parameters, and data types; and a list of errors that the service returns.
<a href="#">Amazon SQS Release Notes</a>	The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
<a href="#">AWS Developer Resource Center</a>	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
<a href="#">Discussion Forums</a>	A community-based forum for developers to discuss technical questions related to Amazon SQS.
<a href="#">AWS Support Center</a>	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and AWS Premium Support (if you are subscribed to this program).
<a href="#">AWS Premium Support Information</a>	The primary web page for information about AWS Premium Support, a one-on-one, fast-response support channel to help you build and run applications on AWS Infrastructure Services.
<a href="#">Product information for Amazon SQS</a>	The primary web page for information about Amazon SQS.
<a href="#">Contact Us</a>	A central contact point for inquiries concerning AWS billing, account, events, abuse etc.
<a href="#">Conditions of Use</a>	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

# Introduction to Amazon SQS

---

## Topics

- [Overview of Amazon SQS \(p. 3\)](#)
- [Features \(p. 4\)](#)
- [Message Lifecycle \(p. 4\)](#)
- [About the Samples \(p. 6\)](#)

This introduction to Amazon SQS is intended to give you a high-level overview of this web service. After reading this section, you should understand the basics you need to work through the examples in this guide.

## Overview of Amazon SQS

Amazon SQS is a distributed queue system that enables web service applications to quickly and reliably queue messages that one component in the application generates to be consumed by another component. A queue is a temporary repository for messages that are awaiting processing.

Using Amazon SQS, you can decouple the components of an application so they run independently, with Amazon SQS easing message management between components. Any component of a distributed application can store messages in a fail-safe queue. Messages can contain up to 64 KiB of text in any format. Any component can later retrieve the messages programmatically using the SQS API.

The queue acts as a buffer between the component producing and saving data, and the component receiving the data for processing. This means the queue resolves issues that arise if the producer is producing work faster than the consumer can process it, or if the producer or consumer are only intermittently connected to the network.

SQS ensures delivery of each message at least once, and supports multiple readers and writers interacting with the same queue. A single queue can be used simultaneously by many distributed application components, with no need for those components to coordinate with each other to share the queue.

Amazon SQS is engineered to always be available and deliver messages. One of the resulting tradeoffs is that SQS does not guarantee first in, first out delivery of messages. For many distributed applications, each message can stand on its own, and as long as all messages are delivered, the order is not important. If your system requires that order be preserved, you can place sequencing information in each message, so that you can reorder the messages when the queue returns them.

## Features

Amazon SQS provides the following major features:

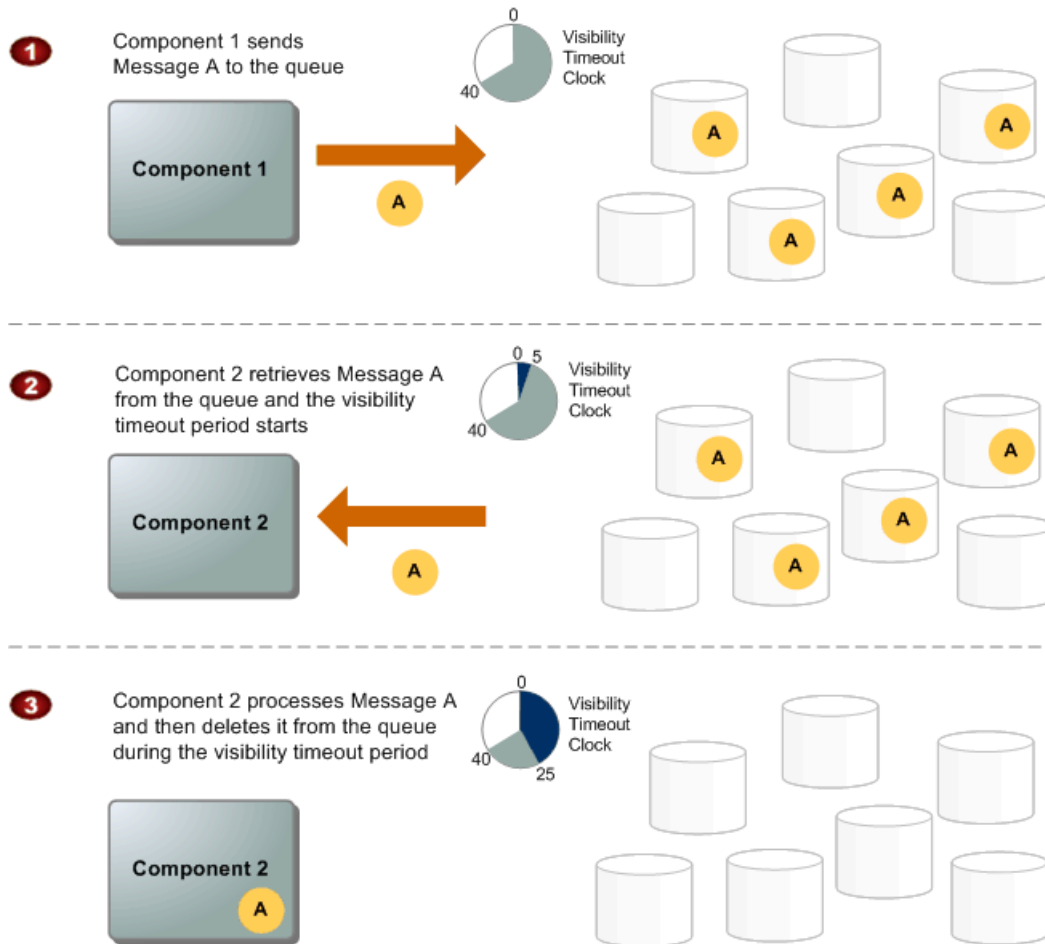
- **Redundant infrastructure**—Guarantees delivery of your messages at least once, highly concurrent access to messages, and high availability for sending and retrieving messages
- **Multiple writers and readers**—Multiple parts of your system can send or receive messages at the same time  
SQS locks the message during processing, keeping other parts of your system from processing the message simultaneously.
- **Configurable settings per queue**—All of your queues don't have to be exactly alike  
For example, one queue can be optimized for messages that require a longer processing time than others.
- **Variable message size**—Your messages can be up to 65536 bytes (64 KiB) in size  
For even larger messages, you can store the contents of the message using the Amazon Simple Storage Service (Amazon S3) or Amazon SimpleDB and use Amazon SQS to hold a pointer to the Amazon S3 or Amazon SDB object. Alternately, you can split the larger message into smaller ones.  
For more information about the services, go to the [Amazon S3 detail page](#) and the [Amazon SimpleDB detail page](#).
- **Access control**—You can control who can send messages to a queue, and who can receive messages from a queue
- **Delay Queues**—A delay queue is one which the user sets a default delay on a queue such that delivery of all messages enqueued will be postponed for that duration of time. You can set the delay value when you create a queue with `CreateQueue`, and you can update the value with `SetQueueAttributes`. If you update the value, the new value affects only messages enqueued after the update.

## Message Lifecycle

The following diagram and process describe the lifecycle of an Amazon SQS message, called *Message A*, from creation to deletion. Assume that a queue already exists.

## Amazon Simple Queue Service Getting Started Guide

### Message Lifecycle



### Message Lifecycle

<b>1</b>	Component 1 sends Message A to a queue and the message is redundantly distributed across the SQS servers.
<b>2</b>	When Component 2 is ready to process a message, it retrieves messages from the queue, and Message A is returned. While Message A is being processed, it remains in the queue and is not returned to subsequent receive requests for the duration of the <a href="#">visibility timeout</a> .
<b>3</b>	Component 2 deletes Message A from the queue to avoid the message being received and processed again once the visibility timeout expires.



### Note

SQS automatically deletes messages that have been in a queue for more than maximum message retention period. The default message retention period is 4 days. However, you can set the message retention period to a value from 60 seconds to 1209600 seconds (14 days) with [SetQueueAttributes](#).



## About the Samples

In the preceding section, we discussed in general terms how your system establishes a queue, confirms it's ready to use, and then starts using it. During use, the various components of your system continually send, receive, and delete messages. The sample libraries available with this guide cover *all* possible operations available with SQS. However, the examples in this guide focus specifically on the core queue operations:

- Creating a queue
- Listing your queues
- Sending a message to a queue
- Retrieving messages from a queue
- Deleting a message from a queue

For more specific information about the samples, see [Preparing the Samples \(p. 13\)](#).

For information about the other operations you can perform with SQS, refer to the [Amazon Simple Queue Service Developer Guide](#).

# Getting Set Up

---

This section walks you through each of the tasks you must complete before you can submit an Amazon SQS request. They are presented in the best order to follow so that you can run the samples as quickly as possible. The following tables shows the sections you need to read if you're already an AWS user, or if you're brand new to AWS.

Existing AWS User	New AWS User
<ol style="list-style-type: none"><li>1. <a href="#">Signing Up for SQS (p. 8)</a></li><li>2. <a href="#">Getting the Tools You Need (p. 11)</a></li></ol>	<ol style="list-style-type: none"><li>1. <a href="#">Creating an AWS Account (p. 7)</a></li><li>2. <a href="#">Signing Up for SQS (p. 8)</a></li><li>3. <a href="#">Getting Your AWS Identifiers (p. 9)</a></li><li>4. <a href="#">Getting the Tools You Need (p. 11)</a></li></ol>

## Creating an AWS Account



### Tip

If you're already an AWS service user, you can skip directly to [Signing Up for SQS \(p. 8\)](#).

To access any web service AWS offers, you must first create an AWS account at <http://aws.amazon.com>. An AWS account is simply an Amazon.com account that is enabled to use AWS products; you can use an existing Amazon.com account login and password when creating the AWS account.



### Important

If you have a personal Amazon.com account, you might want to have a separate Amazon.com account just for your AWS activity. You could provide a new e-mail address not already in the Amazon.com system, or provide an e-mail address for an existing Amazon.com account you have but use a different password. You can have multiple Amazon.com accounts that use the same e-mail address, but different passwords.

From your AWS account you can view your AWS account activity, view usage reports, and manage your AWS account access identifiers.

### To set up a new account

1. Go to <http://aws.amazon.com>.
2. In the **Sign Up for AWS** box, click **Sign up today**.  
The **Sign In** page is displayed.
3. Enter a valid e-mail address, select the button for **No, I am a new customer**, and click **Continue**.  
The next page asks for the name you want associated with the account (e.g., Joe Smith) and a password for the account. If you're using an e-mail address already in the Amazon.com system, the page indicates that this is an existing address, but still lets you create a new account with it.
4. Enter the name you want associated with the account and a password and click **Continue**.  
The **Account Info** page is displayed.
5. Enter your contact information and select how you learned about AWS. Then read the AWS Customer Agreement, select the check box to indicate that you've read and agree to the terms of the customer agreement, and click **Continue**.  
The process is complete and you've created your new AWS account.

At this point, you have an AWS account, but you're not signed up to use Amazon SQS yet. For instructions, see [Signing Up for SQS \(p. 8\)](#).

## Signing Up for SQS

Before you can use SQS, you must sign up to use the service. You must already have an AWS account (see [Creating an AWS Account \(p. 7\)](#)).

### To sign up for SQS

1. Go to the [Amazon SQS](#) page, and then click **Sign Up for This Web Service** on the top right corner of the page.

Your next action depends on whether you already have a credit card associated with the account.

2. Choose one of the following actions.

If you already have a credit card associated with the account:	If you don't have a credit card associated with the account:
<ul style="list-style-type: none"><li>• Review the displayed SQS pricing and credit card information and click <b>Complete Sign Up</b>.</li></ul>	<ol style="list-style-type: none"><li>a. Enter information for a valid credit card and click <b>Continue</b>.</li><li>b. Enter the billing address to use with the credit card and click <b>Continue</b>.</li><li>c. Review the displayed SQS pricing and credit card information you provided and click <b>Complete Sign Up</b>.</li></ol>

AWS sends you a confirmation e-mail.

If you're an experienced AWS user, you can skip directly to [Getting the Tools You Need \(p. 11\)](#). If you're new to AWS, go to [Getting Your AWS Identifiers \(p. 9\)](#).

# Getting Your AWS Identifiers



## Tip

If you already have your Secret Access Key and AWS Access Key ID, you can skip directly to [Getting the Tools You Need \(p. 11\)](#).

For all of the samples, you'll need your [AWS access key identifiers](#), which AWS assigned you when you created your AWS account. Following are your AWS access key identifiers:

- Access Key ID (a 20-character, alphanumeric sequence)  
For example: 022QF06E7MXBSH9DHM02
- Secret Access Key (a 40-character sequence)  
For example: kWcrIUX5JEDGM/LtmEENI/aVmYvHNif5zB+d9+ct



## Caution

Your Secret Access Key is a secret, which only you and AWS should know. It is important to keep it confidential to protect your account. Store it securely in a safe place. Never include it in your requests to AWS, and never e-mail it to anyone. Do not share it outside your organization, even if an inquiry appears to come from AWS or Amazon.com. No one who legitimately represents Amazon will ever ask you for your Secret Access Key.

The Access Key ID is associated with your AWS account. You include it in AWS service requests to identify yourself as the sender of the request.

The Access Key ID is not a secret, and anyone could use your Access Key ID in requests to AWS. To provide proof that you truly are the sender of the request, you also include a digital signature calculated using your Secret Access Key. The sample code handles this for you.

Your Access Key ID and Secret Access Key are displayed to you when you create your AWS account. They are not e-mailed to you. If you need to see them again, you can view them at any time from your AWS account.

## To get your AWS access key identifiers

1. Go to the Amazon Web Services web site at <http://aws.amazon.com>.
2. Click **Account**.  
The **Your Account** section displays.
3. Click **Security Credentials**.
4. Log in to your AWS account.  
The **Security Credentials** page is displayed.
5. Your Access Key ID is displayed in the **Access Identifiers** section of the page.
6. To display your Secret Access Key, click **Show** in the **Secret Access Key** column.

### Access Credentials

There are three types of access credentials used to authenticate your requests to AWS services: (a) access keys, (b) X.509 certificates, and (c) key pairs. Each access credential type is explained below.

Access Keys

X.509 Certificates

Key Pairs

Use access keys to make secure REST or Query protocol requests to any AWS service API. We create one for you when your account is created — see your access key below.

**Your Access Keys**

Created	Access Key ID	Secret Access Key	Status
November 13, 2009	AKIAINGJRRZPAIDTRSQQ	<a href="#">Show</a>	Active ( <a href="#">Make Inactive</a> )

[Create a new Access Key](#)

For your protection, you should never share your secret access keys with anyone. In addition, industry best practice recommends frequent key rotation.

[Learn more about Access Keys](#)

## Getting the Tools You Need

If you want to use the sample code that goes with this guide, you must either install the programming tools listed in this section, or you must use the scratchpad we provide for this tutorial..



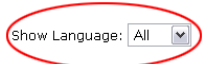
### Note

The Amazon SQS scratchpad is a simple HTML and JavaScript application. If you want to use the scratchpad, skip directly to [Preparing the Samples \(p. 13\)](#) and follow the instructions there to set up and use the scratchpad.



### Tip

In the HTML version of this document, you can hide the sections of this guide that don't apply to the programming language you are using. There is a language selection menu in the upper-right corner of pages with language-specific text. Select your language to hide all others, or select All to show the examples in all available languages.



Show Language: All

Your selection applies to all of the pages in this guide.

## Java

To use the Java sample code, you must have the following tool:

- [Java Standard Edition Development Kit](#)

## C#

To use the C# sample code, you must have the following tools:

- [Microsoft Visual C# 2008 Express Edition](#) or [Microsoft Visual Studio 2008 Professional Edition](#)
- [.NET Framework 2.0 or later](#)

## Perl

To use the Perl sample code, you must have the following tools:

- Perl 5.6.0 or newer (for more information, go to <http://perl.org>)
- The `Digest::SHA1`, `Bundle::LWP`, and `XML::Simple` modules (to download these, go to the [CPAN web site](#))

### To install the required modules

- Run the following commands.

```
perl -MDigest::SHA1 -e 1  
perl -MBundle::LWP -e 1  
perl -MXML::Simple -e 1
```

## PHP5

To use the PHP sample code, you must have the following tool:

- PHP 5.2.1 or newer (for more information, go to <http://php.net>)

# Working with Amazon SQS

---

## Topics

- [Preparing the Samples \(p. 13\)](#)
- [Creating a Queue \(p. 18\)](#)
- [Confirming the Queue Exists \(p. 21\)](#)
- [Sending a Message \(p. 24\)](#)
- [Receiving a Message \(p. 29\)](#)
- [Deleting a Message \(p. 35\)](#)

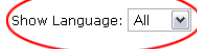
This section describes how to create an Amazon SQS queue, send a message to the queue, and retrieve and delete messages from the queue. The following sections are intended to be followed sequentially, like a tutorial.

You can explore Amazon SQS without writing code by using either the AWS Management Console or the Amazon SQS scratchpad. You can also code with one of the following: Java, C#, Perl, or PHP5.



## Tip

In the HTML version of this document, you can hide the sections of this guide that don't apply to the programming language you are using. There is a language selection menu in the upper-right corner of pages with language-specific text. Select your language to hide all others, or select All to show the examples in all available languages.



Your selection applies to all of the pages in this guide.

## Preparing the Samples

To complete the subsequent sections in this guide, you must make configuration changes to the sample files.





### Tip

If you don't want to write code, you can use the AWS Management Console or Amazon SQS scratchpad, a simple HTML and JavaScript application.

## AWS Management Console

The easiest way to explore Amazon SQS without writing any code is with the AWS Management Console.

### To access SQS with the AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. Select the **Amazon SQS** tab.

## Scratchpad

An easy way to explore Amazon SQS without writing any code is with the Amazon SQS scratchpad. To use the scratchpad, you must have an AWS account and be signed up for SQS (for more information, see [Creating an AWS Account](#) (p. 7) and [Signing Up for SQS](#) (p. 8)).

The scratchpad is a simple HTML and JavaScript application based on the 2009-02-01 version of SQS. You can use the scratchpad to understand what Query requests and responses for SQS look like, and to troubleshoot signature issues with Query requests.

### To prepare the scratchpad

1. Go to the [scratchpad download](#).
2. Download the scratchpad and extract it to a working directory on your computer.
3. Open the `index.html` file with a web browser.  
The welcome page is displayed.



4. Enter your Access Key ID and Secret Access Key (for more information, see [Getting Your AWS Identifiers](#) (p. 9)).
5. Follow the tutorial described in the remainder of this guide.

## Java

Note that the Java example performs all actions in one call: creating a queue, confirming the queue exists, sending a message, receiving a message, and deleting a message.

### To prepare the sample files

1. Go to the [AWS SDK for Java](#) page and download the SDK.
2. Download the `aws-java-sdk-1.0.0.20100226.zip` file and unzip it to a directory designated as `<sqshome>` on your machine.
3. Open the AWS credentials file: `<sqshome>/aws-java-sdk/samples/AmazonSimpleQueueService/AwsCredentials.properties`.
4. Enter your Access Key ID and Secret Access Key:

```
# Fill in your AWS Access Key ID and Secret Access Key
# http://aws.amazon.com/security-credentials
accessKey = <Your AWS Access Key ID>
secretKey = <Your Secret Access Key>
```

5. Save the file.

## C#

Note that the C# example performs all actions in one call: creating a queue, confirming the queue exists, sending a message, receiving a message, and deleting a message.

### To prepare the sample files

1. Go to the [AWS SDK for .NET](#) page and download the SDK.
2. Run the downloaded installation program, `AWSToolsAndSDKForNet.msi`.  
The installation installs to `C:\Program Files\AWS SDK for .NET` by default.
3. Go to `C:\Program Files\AWS SDK for .NET\Samples\AmazonSQS_Sample` and open one of the following sample solutions:
  - `AmazonSQS_Sample.VS2008.sln`
  - `AmazonSQS_Sample.VS2010.sln`
4. Open the `App.config` file.
5. Enter your Access Key ID and Secret Access Key:

```
<configuration>
  <appSettings>
    <add key="AWSAccessKey" value="<Your Access Key ID>" />
    <add key="AWSSecretKey" value="<Your Secret Access Key>" />
  </appSettings>
</configuration>
```

6. Save the file.

## Perl

### To prepare the sample files

1. Go to the location of the [Perl sample code](#) that AWS provides and download the sample code package.
2. Extract the sample code into a directory on your machine.
3. Prepare each of the individual samples:
  - a. Open the  
amazon-queue-2009-02-01-perl-library/src/Amazon/SQS/Samples/CreateQueueSample.pl  
sample file.
  - b. Make sure the following lines are not commented out and enter your Access Key ID and Secret Access Key.

```
my $AWS_ACCESS_KEY_ID      = "<Your Access Key ID>";  
my $AWS_SECRET_ACCESS_KEY  = "<Your Secret Access Key>";  
use Amazon::SQS::Client;  
my $service = Amazon::SQS::Client->new($AWS_ACCESS_KEY_ID, $AWS_SECRET_ACCESS_KEY);
```

Make sure the following lines are commented out.

```
use Amazon::SQS::Mock;  
my $service = Amazon::SQS::Mock->new;
```

- c. Save the file.
- d. Repeat this procedure for each file in the `Samples` directory.



#### Note

The mock service is an alternate way to use the sample code. The service doesn't call AWS, but instead returns a set response that you can modify to suit your needs (the XML response files are in the `Mock` directory). The mock service makes it easy for you to test how your application handles different responses.

## PHP5

### To prepare the sample files

1. Go to the location of the [PHP sample code](#) that AWS provides and download the sample code package.
2. Extract the sample code into a directory on your machine.
3. Open the  
amazon-queue-2009-02-01-php5-library/src/Amazon/SQS/Samples/.config.inc.php  
file.
4. Enter your Access Key ID and Secret Access Key in the following lines and save the file.

```
define('AWS_ACCESS_KEY_ID', '<Your Access Key ID>');  
define('AWS_SECRET_ACCESS_KEY', '<Your Secret Access Key>');
```

5. Prepare each of the individual samples:

- a. Open the `amazon-queue-2009-02-01-php5-library/src/Amazon/SQS/Samples/CreateQueueSample.php` file.
- b. Make sure the following line is not commented out.

```
$service = new Amazon_SQS_Client(AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY);
```

Make sure the following line is commented out.

```
$service = new Amazon_SQS_Mock();
```

- c. Save the file.
- d. Repeat this procedure for each file in the `Samples` directory.



#### Note

The mock service is an alternate way to use the sample code. The service doesn't call AWS, but instead returns a set response that you can modify to suit your needs (the XML response files are in the `Mock` directory). The mock service makes it easy for you to test how your application handles different responses.

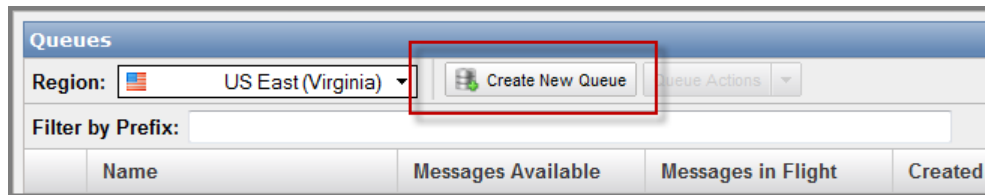
## Creating a Queue

The first task in using Amazon SQS is to create one or more queues. The following examples demonstrate creation of a queue named `MyQueue`.

### AWS Management Console

#### To run the sample

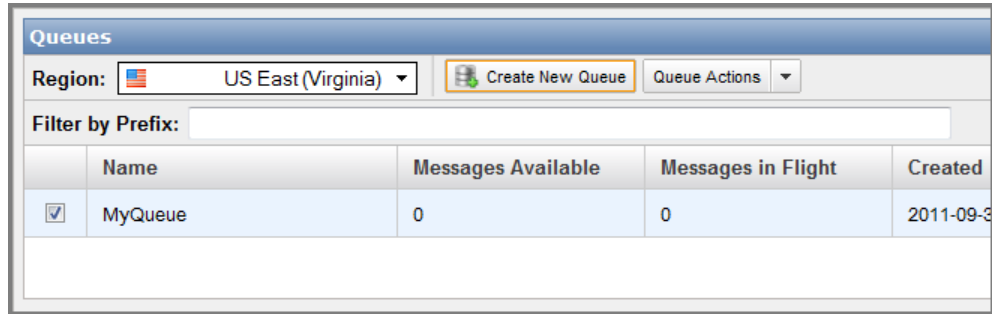
1. In the AWS Management Console click **Create New Queue**.



2. In the **Create New Queue** dialog box, enter `MyQueue` in the **Queue Name** field, and leave the default value settings for the remaining fields.

A screenshot of the 'Create New Queue' dialog box. At the top, it says 'Please enter a name for your new queue. Queue names must be 1-80 characters in length and be composed of alphanumeric characters, hyphens (-), and underscores (\_). Your queue will be created in the US East (Virginia) region.' Below this, the 'Region' is set to 'US East (Virginia)'. The 'Queue Name' field is highlighted with a red rectangle and contains the text 'MyQueue'. Below the queue name field, there's a section titled 'Configure your new queue by setting queue attributes (optional)'. It contains four rows of attributes: 'Default Visibility Timeout' (30 seconds), 'Message Retention Period' (4 days), 'Maximum Message Size' (64 KB), and 'Delivery Delay' (0 seconds). Each attribute has a text input, a unit dropdown, and a descriptive note. At the bottom right, there are 'Cancel' and 'Create Queue' buttons.

3. Click **Create Queue**.  
Your new queue appears in the list of queues.



## Scratchpad

### To run the sample

1. In the scratchpad, select **CreateQueue** from the **Explore API** list box.
2. Enter `MyQueue` in the **Queue Name** field, and leave the **Default Visibility Timeout** field blank.
3. Select one of the following:
  - To invoke the request, click **Invoke Request**. Amazon SQS returns a response.
  - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser. Amazon SQS returns a response.
  - To view the string to sign, click **Display String to Sign**.

## Java

### To run the sample

1. Open `SimpleQueueServiceSample.java`.

The following section of the code creates a queue:

```
// Create a queue
System.out.println("Creating a new SQS queue called MyQueue.\n");
CreateQueueRequest createQueueRequest = new CreateQueueRe
quest().withQueueName("MyQueue");
String myQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();
```

2. Compile and run the sample.  
The `MyQueue` queue is created.

## C#

### To run the sample

1. Open `Program.cs`.

The following section of the code creates a queue:

```
//Creating a queue
Console.WriteLine("Create a queue called MyQueue.\n");
CreateQueueRequest sqsRequest = new CreateQueueRequest();
sqsRequest.QueueName = "MyQueue";
CreateQueueResponse createQueueResponse = sqs.CreateQueue(sqsRequest);
String myQueueUrl;
myQueueUrl = createQueueResponse.CreateQueueResult.QueueUrl;
```

2. Run the sample.  
The MyQueue queue is created.

## Perl

### To run the sample

1. Open CreateQueueSample.pl.
2. Locate the following line.

```
# invokeCreateQueue($service, $request);
```

3. Replace the line with the following new lines of code.

```
my $request = Amazon::SQS::Model::CreateQueueRequest->new( {
    QueueName => "MyQueue"
});
invokeCreateQueue($service, $request);
```

4. Run the sample.  
The MyQueue queue is created.

## PHP5

### To run the sample

1. Open CreateQueueSample.php.
2. Locate the following line.

```
// invokeCreateQueue($service, $request);
```

3. Replace the line with the following new lines of code.

```
require_once ('Amazon/SQS/Model/CreateQueueRequest.php');
$request = new Amazon_SQS_Model_CreateQueueRequest();
$request->setQueueName('MyQueue');
invokeCreateQueue($service, $request);
```

4. Run the sample.  
The MyQueue queue is created.

## Confirming the Queue Exists

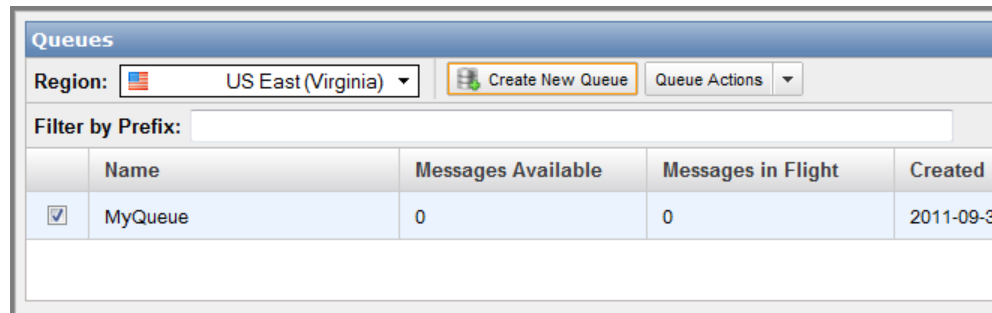
When you create a queue, it can take a short time for the queue to propagate throughout the SQS system. You can confirm the queue's existence by listing the queues you have in SQS. The following code snippets list the queues you've created using the 2009-02-01 version of SQS.

### AWS Management Console

The AWS Management Console displays a list of your queues for the region you have selected. By default, the console displays the US East region.

#### To display a queue list for a specific region

- Select a region from the **Region** drop-down list.  
The console displays all of your queues in that region.



### Scratchpad

#### To run the sample

1. In the scratchpad, select **ListQueues** from the **Explore API** list box.
2. Leave the **Queue Name Prefix** field blank.
3. Select one of the following:
  - To invoke the request, click **Invoke Request**. Amazon SQS returns a response.
  - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser. Amazon SQS returns a response.
  - To view the string to sign, click **Display String to Sign**.

### Java

#### To run the sample

1. Open `SimpleQueueServiceSample.java`.

The following section of the code lists your queues:

```
// List queues
System.out.println("Listing all queues in your account.\n");
```



```
for (String queueUrl : sqs.listQueues().getQueueUrls()) {  
    System.out.println(" QueueUrl: " + queueUrl);  
}  
System.out.println();
```

2. Compile and run the sample.

Amazon SQS returns the list of the queues you've created using the 2009-02-01 version of Amazon SQS, including the newly created `MyQueue` queue. Each queue is identified by its [queue URL](#). The response also includes the request ID Amazon SQS assigned to your request.

## C#

### To run the sample

1. Open `Program.cs`.

The following section of the code lists your queues:

```
//Confirming the queue exists  
ListQueuesRequest listQueuesRequest = new ListQueuesRequest();  
ListQueuesResponse listQueuesResponse = sqs.ListQueues(listQueuesRequest);  
  
Console.WriteLine("Printing list of Amazon SQS queues.\n");  
if (listQueuesResponse.IsSetListQueuesResult())  
{  
    ListQueuesResult listQueuesResult = listQueuesResponse.ListQueuesResult;  
  
    foreach (String queueUrl in listQueuesResult.QueueUrl)  
    {  
        Console.WriteLine(" QueueUrl: {0}", queueUrl);  
    }  
}  
Console.WriteLine();
```

2. Run the sample.

Amazon SQS returns the list of the queues you've created using the 2009-02-01 version of Amazon SQS, including the newly created `MyQueue` queue. Each queue is identified by its [queue URL](#). The response also includes the request ID Amazon SQS assigned to your request.

## Perl

### To run the sample

1. Open `ListQueuesSample.pl`.
2. Locate the following line.

```
# invokeListQueues($service, $request);
```

3. Replace the line with the following new lines of code.

```
my $request = Amazon::SQS::Model::ListQueuesRequest->new();  
invokeListQueues($service, $request);
```

4. Run the sample.

Amazon SQS returns the list of the queues you've created using the 2009-02-01 version of Amazon SQS, including the newly created `MyQueue` queue. Each queue is identified by its [queue URL](#). The response also includes the request ID Amazon SQS assigned to your request.

## PHP5

### To run the sample

1. Open `ListQueuesSample.php`.
2. Locate the following line.

```
// invokeListQueues($service, $request);
```

3. Replace the line with the following new lines of code.

```
require_once ('Amazon/SQS/Model/ListQueuesRequest.php');  
$request = new Amazon_SQS_Model_ListQueuesRequest();  
invokeListQueues($service, $request);
```

4. Run the sample.

Amazon SQS returns the list of the queues you've created using the 2009-02-01 version of Amazon SQS, including the newly created `MyQueue` queue. Each queue is identified by its [queue URL](#). The response also includes the request ID Amazon SQS assigned to your request.

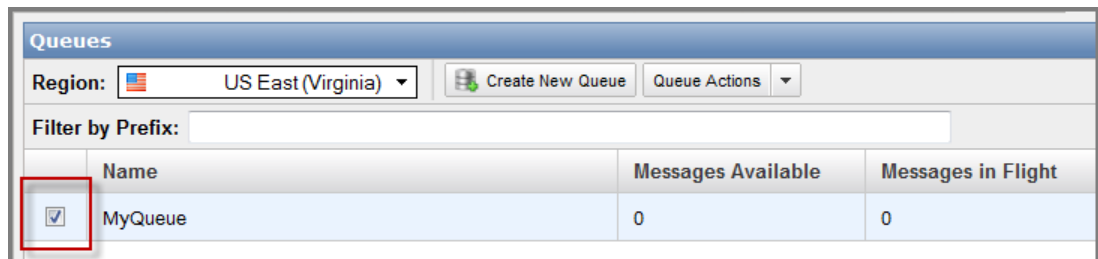
## Sending a Message

Now that you've confirmed your queue exists in the Amazon SQS system, you can send a message to the queue. The following code snippets demonstrate how to send the message `This is my message text.` to your `MyQueue` queue.

### AWS Management Console

#### To send a message

1. In the AWS Management Console select a queue.

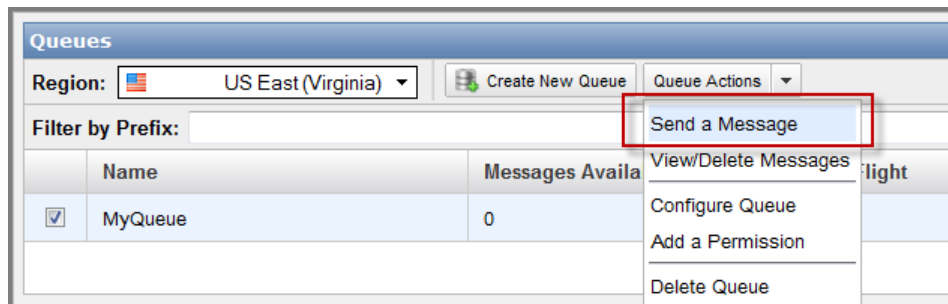


2. Select **Send a Message** from the **Queue Actions** drop-down list.

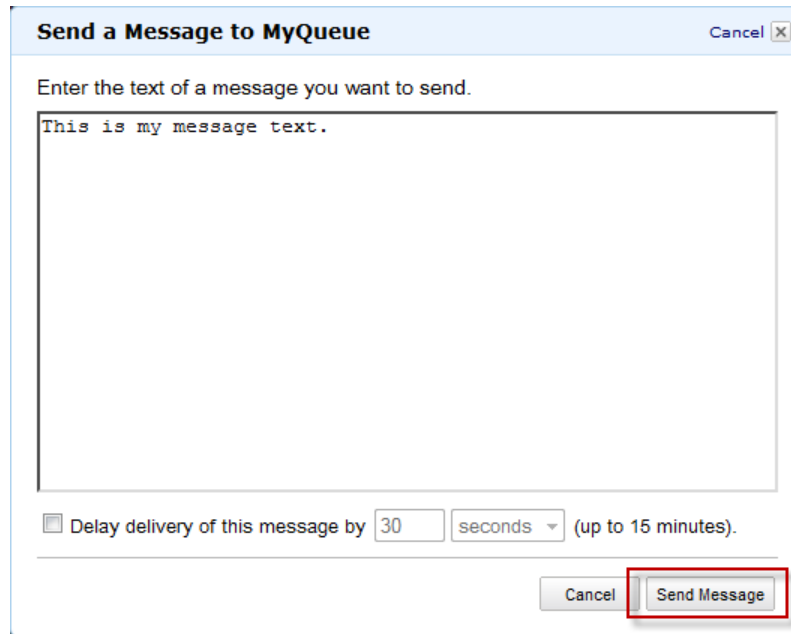


#### Note

The **Queue Actions** drop-down list is available only if a queue is selected.

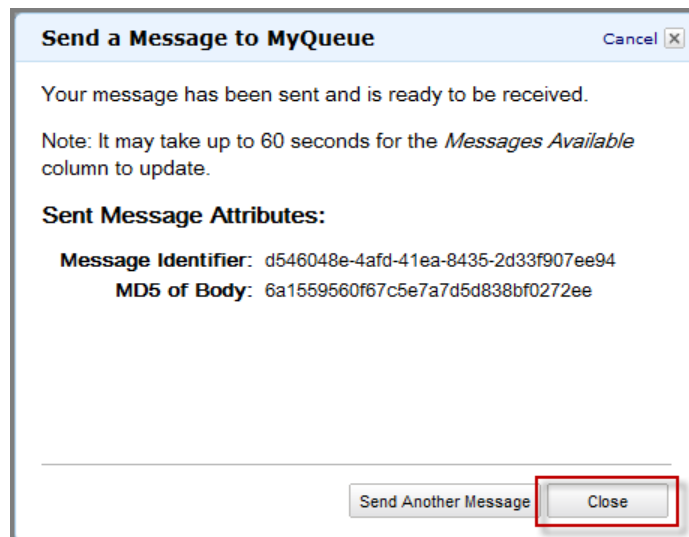


3. In the **Send a Message to MyQueue** dialog box, enter `This is my message text.` and click **Send Message**.



The dialog box titled "Send a Message to MyQueue" has a "Cancel" button in the top right corner. Below the title bar, it says "Enter the text of a message you want to send." There is a large text area containing the text "This is my message text." Below the text area, there is a checkbox labeled "Delay delivery of this message by" followed by a text input field containing "30", a dropdown menu showing "seconds", and the text "(up to 15 minutes)". At the bottom right, there are two buttons: "Cancel" and "Send Message". The "Send Message" button is highlighted with a red rectangle.

4. In the **Send a Message to MyQueue** confirmation box click **Close**.



The confirmation dialog box titled "Send a Message to MyQueue" has a "Cancel" button in the top right corner. The main text says "Your message has been sent and is ready to be received." Below this, a note states: "Note: It may take up to 60 seconds for the *Messages Available* column to update." Under the heading "Sent Message Attributes:", there are two lines of text: "Message Identifier: d546048e-4afd-41ea-8435-2d33f907ee94" and "MD5 of Body: 6a1559560f67c5e7a7d5d838bf0272ee". At the bottom, there are two buttons: "Send Another Message" and "Close". The "Close" button is highlighted with a red rectangle.

## Scratchpad

### To run the sample

1. In the scratchpad, select **SendMessage** from the **Explore API** list box.
2. Enter the queue URL (which you received when you created the queue) in the **Queue URL** field.
3. Enter `This is my message text.` in the **Message Body** field.
4. Select one of the following:
  - To invoke the request, click **Invoke Request**. Amazon SQS returns a response.

- To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser. Amazon SQS returns a response.
- To view the string to sign, click **Display String to Sign**.

## Java

### To run the sample

1. Open `SimpleQueueServiceSample.java`.

The following section of the code sends a message to your queue:

```
// Send a message
System.out.println("Sending a message to MyQueue.\n");
sqs.sendMessage(new SendMessageRequest()
    .withQueueUrl(myQueueUrl)
    .withMessageBody("This is my message text."));
```

2. Compile and run the sample.

The message `This is my message text.` is sent to the queue. The response includes the following items:

- The [message ID](#) Amazon SQS assigns to the message
- An MD5 digest of the message body, which you can use to confirm that SQS received the message correctly (for information about MD5, go to <http://faqs.org/rfcs/rfc1321.html>)
- The request ID that Amazon SQS assigned to your request

## C#

### To run the sample

1. Open `Program.cs`.

The following section of the code sends a message to your queue:

```
//Sending a message
Console.WriteLine("Sending a message to MyQueue.\n");
SendMessageRequest sendMessageRequest = new SendMessageRequest();
sendMessageRequest.QueueUrl = myQueueUrl; //URL from initial queue creation
sendMessageRequest.MessageBody = "This is my message text.";
sqs.SendMessage(sendMessageRequest);
```

2. Run the sample.

The message `This is my message text.` is sent to the queue. The response includes the following items:

- The [message ID](#) Amazon SQS assigns to the message
- An MD5 digest of the message body, which you can use to confirm that SQS received the message correctly (for information about MD5, go to <http://faqs.org/rfcs/rfc1321.html>)
- The request ID that Amazon SQS assigned to your request

## Perl

### To run the sample

1. Open `SendMessageSample.pl`.
2. Locate the following line.

```
# invokeSendMessage($service, $request);
```

3. Replace the line with the following new lines of code.

```
my $request = Amazon::SQS::Model::SendMessageRequest->new({  
  QueueUrl => "queue URL you received from CreateQueue call",  
  MessageBody => "This is my message text."  
});  
invokeSendMessage($service, $request);
```

4. Run the sample.  
The message `This is my message text.` is sent to the queue. The response includes the following items:
  - The [message ID](#) Amazon SQS assigns to the message
  - An MD5 digest of the message body, which you can use to confirm that SQS received the message correctly (for information about MD5, go to <http://faqs.org/rfcs/rfc1321.html>)
  - The request ID that Amazon SQS assigned to your request

## PHP5

### To run the sample

1. Open `SendMessageSample.php`.
2. Locate the following line.

```
// invokeSendMessage($service, $request);
```

3. Replace the line with the following new lines of code.

```
require_once ('Amazon/SQS/Model/SendMessageRequest.php');  
$request = new Amazon_SQS_Model_SendMessageRequest();  
$request->setQueueUrl('queue URL you received from CreateQueue call');  
$request->setMessageBody('This is my message text.');
```

```
invokeSendMessage($service, $request);
```

4. Run the sample.  
The message `This is my message text.` is sent to the queue. The response includes the following items:
  - The [message ID](#) Amazon SQS assigns to the message
  - An MD5 digest of the message body, which you can use to confirm that SQS received the message correctly (for information about MD5, go to <http://faqs.org/rfcs/rfc1321.html>)

- The request ID that Amazon SQS assigned to your request

## Receiving a Message

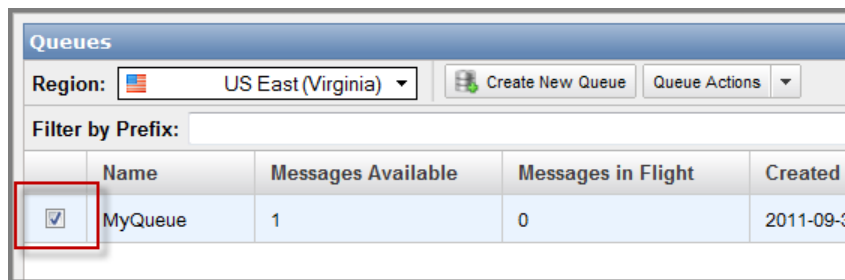
Now that a message is in the queue, you can receive it (retrieve it from the queue). When requesting to get a message from the queue, you can't specify which message to get. You simply specify the maximum number of messages you want to get (up to 10), and SQS returns up to that maximum number. Because SQS is a distributed system and the particular queue we're working with here has very few messages in it, the response to the receive request might be empty. Therefore, you should rerun the sample until you get the message. You should design your own application so that it continues to poll until it gets one or more messages.

Amazon SQS doesn't automatically delete the message after returning it to you, in case you don't actually receive the message (the receiving component could fail or lose its connection). You must send a separate request to delete the message, which acknowledges that you've successfully received and processed the message. For more information, see [Deleting a Message](#) (p. 35).

## AWS Management Console

### To receive a message

1. In the AWS Management Console select a queue.

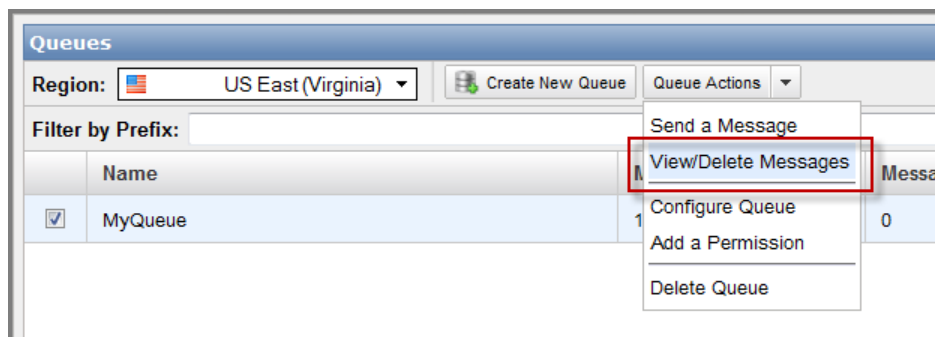


2. Select **View/Delete Messages** from the **Queue Actions** drop-down list.



#### Note

The **Queue Actions** drop-down list is available only if a queue is selected.



3. Click **Start Polling for Messages** to receive a message from the queue.



View messages currently available in the queue by clicking the *Start Polling for Messages* button.

- Messages will come from the front of the queue unless other applications are also reading from the queue.
- Messages displayed in the console will not be available to other applications until the console stops polling for messages.
- The console will stop polling for messages as soon as the specified number of seconds have elapsed, the requested number of messages have been received, or the *Stop Now* button has been pressed.
- Deleting a message from the console permanently removes it from the queue.

☐ Don't show this again.

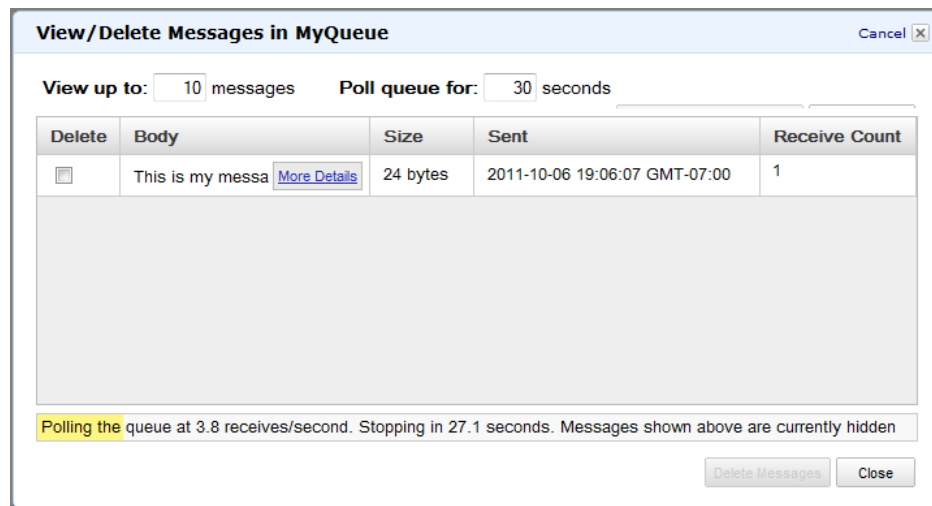
**Start Polling for Messages** 



### Note

The **Start Polling for Messages** dialog box will not appear if you have previously selected the **Don't show this again** checkbox.

The **View/Delete Messages in MyQueue** dialog box displays a message from the queue.

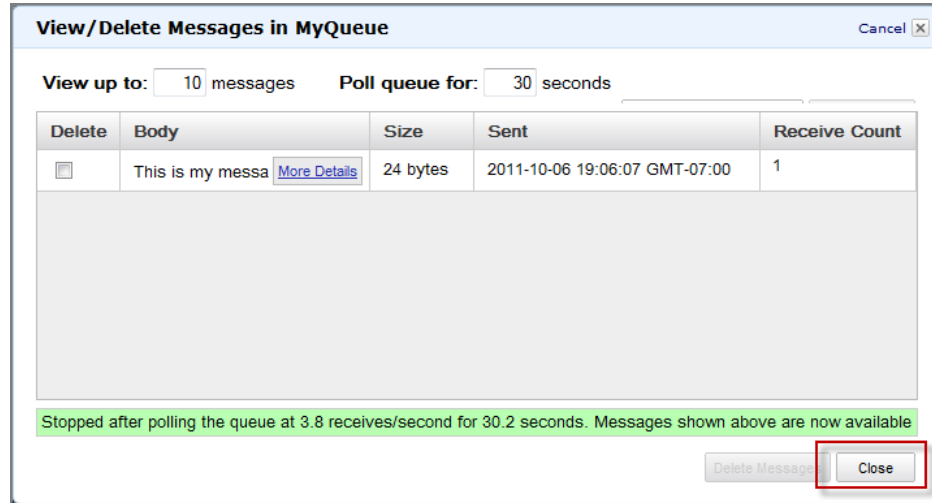


Delete	Body	Size	Sent	Receive Count
<input type="checkbox"/>	This is my messa <a href="#">More Details</a>	24 bytes	2011-10-06 19:06:07 GMT-07:00	1

Polling the queue at 3.8 receives/second. Stopping in 27.1 seconds. Messages shown above are currently hidden

Delete Messages Close

A yellow progress bar at the bottom of the dialog box displays the status of the message's visibility timeout. While the bar is yellow, the message is not visible to other consumers. When the bar turns green, the visibility timeout is complete and the message is once again visible to other consumers.



4. Click **Close** to close the **View/Delete Messages in MyQueue** dialog box.

## Scratchpad

### To run the sample

1. In the scratchpad, select **ReceiveMessage** from the **Explore API** list box.
2. Enter the queue URL in the **Queue URL** field.
3. Leave the **Max Number Of Messages** and the **Visibility Timeout** fields blank.
4. Select one of the following:
  - To invoke the request, click **Invoke Request**. Amazon SQS returns a response.
  - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser. Amazon SQS returns a response.
  - To view the string to sign, click **Display String to Sign**.

## Java

### To run the sample

1. Open `SimpleQueueServiceSample.java`.

The following section of the code receives a message from your queue:

```
System.out.println("Receiving messages from MyQueue.\n");
ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRe
quest(myQueueUrl);
List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMes
sages();
for (Message message : messages) {
    System.out.println("    Message");
    System.out.println("        MessageId:      " + message.getMessageId());
    System.out.println("        ReceiptHandle: " + message.getReceiptHandle());
    System.out.println("        MD5OfBody:      " + message.getMD5OfBody());
```

```
System.out.println("    Body:          " + message.getBody());
for (Entry<String, String> entry : message.getAttributes().entrySet()) {

    System.out.println("    Attribute");
    System.out.println("        Name:  " + entry.getKey());
    System.out.println("        Value: " + entry.getValue());
}
}
System.out.println();
```

2. Compile and run the sample.

The `MyQueue` queue is polled for messages and returns 0 or more messages. The sample prints the following items:

- The [message ID](#) that you received when you sent the message to the queue
- The [receipt handle](#) (which you use later to delete the message)
- An MD5 digest of the message body (for information about MD5, go to <http://faqs.org/rfcs/rfc1321.html>)
- The message body
- The request ID that Amazon SQS assigned to your request

If no messages are received in this particular call, the response includes only the request ID.

## C#

### To run the sample

1. Open `Program.cs`.

The following section of the code receives a message from your queue:

```
//Receiving a message
ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest();
receiveMessageRequest.QueueUrl = myQueueUrl;
ReceiveMessageResponse receiveMessageResponse = sqs.ReceiveMessage(receiveMessageRequest);
if (receiveMessageResponse.IsSetReceiveMessageResult())
{
    Console.WriteLine("Printing received message.\n");
    ReceiveMessageResult receiveMessageResult = receiveMessageResponse.ReceiveMessageResult;
    foreach (Message message in receiveMessageResult.Message)
    {
        Console.WriteLine("    Message");
        if (message.IsSetMessageId())
        {
            Console.WriteLine("        MessageId: {0}", message.MessageId);
        }
        if (message.IsSetReceiptHandle())
        {
            Console.WriteLine("        ReceiptHandle: {0}", message.ReceiptHandle);
        }
        if (message.IsSetMD5OfBody())
        {
            Console.WriteLine("        MD5OfBody: {0}", message.MD5OfBody);
        }
    }
}
```

```
}
if (message.IsSetBody())
{
    Console.WriteLine("    Body: {0}", message.Body);
}
foreach (Amazon.SQS.Model.Attribute attribute in message.Attribute)
{
    Console.WriteLine("  Attribute");
    if (attribute.IsSetName())
    {
        Console.WriteLine("    Name: {0}", attribute.Name);
    }
    if (attribute.IsSetValue())
    {
        Console.WriteLine("    Value: {0}", attribute.Value);
    }
}
}
}
String messageReceiptHandle = receiveMessageResponse.ReceiveMessageResult.Mes
sage[0].ReceiptHandle;
```

2. Run the sample.

The `MyQueue` queue is polled for messages and returns 0 or more messages. The sample prints the following items:

- The [message ID](#) that you received when you sent the message to the queue
- The [receipt handle](#) (which you use later to delete the message)
- An MD5 digest of the message body (for information about MD5, go to <http://faqs.org/rfcs/rfc1321.html>)
- The message body
- The request ID that Amazon SQS assigned to your request

If no messages are received in this particular call, the response includes only the request ID.

## Perl

### To run the sample

1. Open `ReceiveMessageSample.pl`.
2. Locate the following line.

```
# invokeReceiveMessage($service, $request);
```

3. Replace the line with the following new lines of code.

```
my $request = Amazon::SQS::Model::ReceiveMessageRequest->new({
    QueueUrl => "queue URL you received from CreateQueue call"
});
invokeReceiveMessage($service, $request);
```

4. Run the sample.

The `MyQueue` queue is polled for messages and returns 0 or more messages. The sample prints the following items:

- The [message ID](#) that you received when you sent the message to the queue
- The [receipt handle](#) (which you use later to delete the message)
- An MD5 digest of the message body (for information about MD5, go to <http://faqs.org/rfcs/rfc1321.html>)
- The message body
- The request ID that Amazon SQS assigned to your request

If no messages are received in this particular call, the response includes only the request ID.

## PHP5

### To run the sample

1. Open `ReceiveMessageSample.php`.
2. Locate the following line.

```
// invokeReceiveMessage($service, $request);
```

3. Replace the line with the following new lines of code.

```
require_once ('Amazon/SQS/Model/ReceiveMessageRequest.php');  
$request = new Amazon_SQS_Model_ReceiveMessageRequest();  
$request->setQueueUrl('queue URL you received from CreateQueue call');  
invokeReceiveMessage($service, $request);
```

4. Run the sample.

The `MyQueue` queue is polled for messages and returns 0 or more messages. The sample prints the following items:

- The [message ID](#) that you received when you sent the message to the queue
- The [receipt handle](#) (which you use later to delete the message)
- An MD5 digest of the message body (for information about MD5, go to <http://faqs.org/rfcs/rfc1321.html>)
- The message body
- The request ID that Amazon SQS assigned to your request

If no messages are received in this particular call, the response includes only the request ID.

## Deleting a Message

Once you receive the message, you must delete it from the queue to acknowledge that you processed the message and no longer need it. You specify which message to delete by providing the [receipt handle](#) that Amazon SQS returned when you received the message. You can delete only one message per call. You can delete an entire queue with a call to `DeleteQueue`, even if the queue has messages in it.



### Note

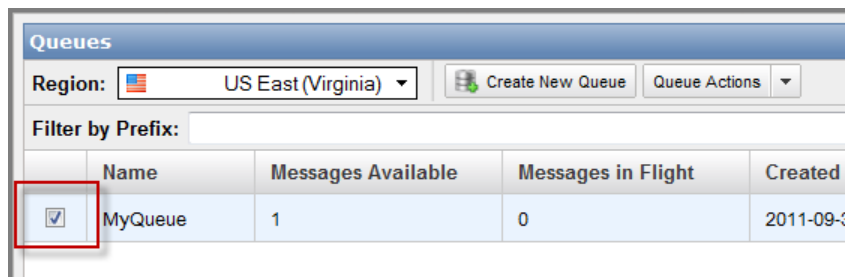
If you don't have the receipt handle for the message, you can call `ReceiveMessage` again and receive the message again. Each time you receive the message, you get a different receipt handle. Use the latest receipt handle when calling `DeleteMessage`, otherwise your message might not be deleted from the queue.

The following examples demonstrate how to delete the message from your `MyQueue` queue.

## AWS Management Console

### To delete a message

1. In the AWS Management Console select a queue.

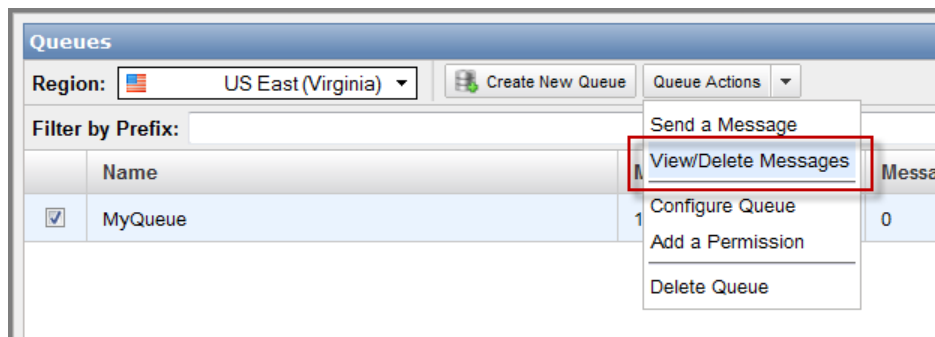


2. Select **View/Delete Messages** from the **Queue Actions** drop-down list.



### Note

The **Queue Actions** drop-down list is available only if a queue is selected.



3. Click **Start Polling for Messages** to view a message from the queue.

View messages currently available in the queue by clicking the *Start Polling for Messages* button.

- Messages will come from the front of the queue unless other applications are also reading from the queue.
- Messages displayed in the console will not be available to other applications until the console stops polling for messages.
- The console will stop polling for messages as soon as the specified number of seconds have elapsed, the requested number of messages have been received, or the *Stop Now* button has been pressed.
- Deleting a message from the console permanently removes it from the queue.

☐ Don't show this again.


**Start Polling for Messages** 



### Note

The **Start Polling for Messages** dialog box will not appear if you have previously selected the **Don't show this again** checkbox.

The **View/Delete Messages** dialog box displays a message from the queue.

**View/Delete Messages in MyQueue** Cancel 

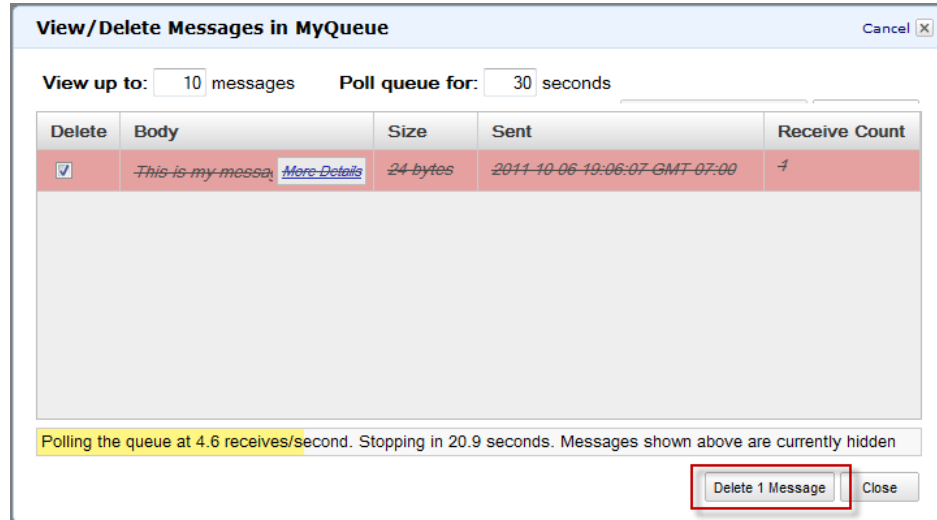
**View up to:**  messages    **Poll queue for:**  seconds

Delete	Body	Size	Sent	Receive Count
<input type="checkbox"/>	This is my messa <a href="#">More Details</a>	24 bytes	2011-10-06 19:06:07 GMT-07:00	1

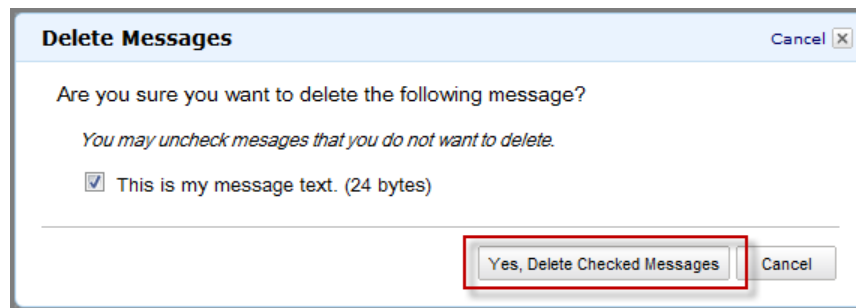
**Polling the** queue at 3.8 receives/second. Stopping in 27.1 seconds. Messages shown above are currently hidden

Delete Messages Close

4. Select the message you want to delete.



- Click **Delete 1 Message** to delete the selected message.  
A **Delete Messages** confirmation dialog box appears.



- Click **Yes, Delete Checked Messages**.  
The selected message is deleted.
- Click **Close** to close the **View/Delete Messages** dialog box.

## Scratchpad

### To run the sample

- In the scratchpad, select **DeleteMessage** from the **Explore API** list box.
- Enter the queue URL in the **Queue URL** field.
- Enter the receipt handle (which you received from the preceding `ReceiveMessage` call) in the **Receipt Handle** field.
- Select one of the following:
  - To invoke the request, click **Invoke Request**. Amazon SQS returns a response.
  - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser. Amazon SQS returns a response.
  - To view the string to sign, click **Display String to Sign**.



## Java

### To run the sample

1. Open `SimpleQueueServiceSample.java`.

The following section of the code deletes a message:

```
// Delete a message
System.out.println("Deleting a message.\n");
String messageReceiptHandle = messages.get(0).getReceiptHandle();
sqs.deleteMessage(new DeleteMessageRequest()
    .withQueueUrl(myQueueUrl)
    .withReceiptHandle(messageReceiptHandle));
```

2. Compile and run the sample.

The message is deleted from the `MyQueue` queue. The response includes the request ID that Amazon SQS assigned to your request.

## C#

### To run the sample

1. Open `Program.cs`.

The following section of the code deletes a message:

```
//Deleting a message
Console.WriteLine("Deleting the message.\n");
DeleteMessageRequest deleteRequest = new DeleteMessageRequest();
deleteRequest.QueueUrl = myQueueUrl;
deleteRequest.ReceiptHandle = messageReceiptHandle;
sqs.DeleteMessage(deleteRequest);
```

2. Run the sample.

The message is deleted from the `MyQueue` queue. The response includes the request ID that Amazon SQS assigned to your request.

## Perl

### To run the sample

1. Open `DeleteMessageSample.pl`.
2. Locate the following line.

```
# invokeDeleteMessage($service, $request);
```

3. Replace the line with the following new lines of code.

```
my $request = Amazon::SQS::Model::DeleteMessageRequest->new({
    QueueUrl => "queue URL you received from CreateQueue call",
    ReceiptHandle => "Receipt handle you received from ReceiveMessage call"
});
invokeDeleteMessage($service, $request);
```

4. Run the sample.

The message is deleted from the `MyQueue` queue. The response includes the request ID that Amazon SQS assigned to your request.

## PHP5

### To run the sample

1. Open `DeleteMessageSample.php`.
2. Locate the following line.

```
// invokeDeleteMessage($service, $request);
```

3. Replace the line with the following new lines of code.

```
require_once ('Amazon/SQS/Model/DeleteMessageRequest.php');
$request = new Amazon_SQS_Model_DeleteMessageRequest();
$request->setQueueUrl('queue URL you received from CreateQueue call');
$request->setReceiptHandle('Receipt handle you received from ReceiveMessage call');
invokeDeleteMessage($service, $request);
```

4. Run the sample.

The message is deleted from the `MyQueue` queue. The response includes the request ID that Amazon SQS assigned to your request.

## You're Finished!

Congratulations! You successfully created a queue, and sent, received, and deleted a message. For information on ideas that were not covered in this guide and how to continue, see [Where Do I Go from Here?](#) (p. 40).

## Please Give Us Your Feedback

Your input is important to us to help make our documentation helpful and easy to use. Please take a minute to give us your feedback on how well we were able to help you get started with Amazon SQS. Just click this [Feedback Link](#) link. Thank you.

## Where Do I 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 SQS, and where to go in the Amazon SQS Developer Guide for more information and instructions. This section describes the next steps we recommend you take.

### Look at the Articles and Tutorials

We recommend that you read the articles and tutorials available on the [Amazon SQS Resource Center](#). Specifically, there are two webcasts designed to help you gain a broader understanding of Amazon SQS:

- [Amazon SQS: The Queue as Glue](#) is a brief presentation that shows how Amazon SQS integrates with other elements in a service-oriented architecture (SOA) application.
- [Use Amazon SQS to Build Self-Healing Applications](#) presents the concepts for developing an application using Amazon SQS as a workflow queue. These ideas help ensure that all processes within a workflow get completed.

### Read the Forum

We recommend you look at the [Amazon SQS 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 this service.

### Look at Other Available Sample Code

You're already aware of the sample code that goes with this guide (for more information, see [Preparing the Samples \(p. 13\)](#)). You can look at other sample code that's available from the [Sample Code and Libraries](#) page, or visit the Developer Center for a specific programming language:

- [Java Developer Center](#)
- [PHP Developer Center](#)
- [Ruby Developer Center](#)
- [Windows & .NET Developer Center](#)

## AWS Account and Security Credentials

So far you signed up for the service, got an AWS account and security credentials, and then completed a short exercise covering the essential product functions. Now that you're finished with the exercise, we recommend that you check with an administrator or coworker in your organization to determine if he or she already has an AWS account and security credentials for you to use in future interactions with AWS.

If you're an account owner or administrator and want to know more about AWS Identity and Access Management, go to the product description at <http://aws.amazon.com/iam> or to the technical documentation at [Using AWS Identity and Access Management](#).

# Document History

---

This documentation is associated with the 2011-10-01 release of the Amazon Simple Queue Service. This guide was last updated on 29 December 2011.

The following table describes the important changes since the last release of the *Amazon Simple Queue Service Getting Started Guide*.

Change	Description	Release Date
New feature	Amazon SQS is now available through the <a href="#">AWS Management Console</a> . For an example of how to create a queue, send a message, and receive a message with the AWS Management Console, see <a href="#">Working with Amazon SQS (p. 13)</a> .	
New feature	This service now integrates with AWS Identity and Access Management (IAM). For more information, go to Integrating with Other AWS Products in <a href="#">Using AWS Identity and Access Management</a> .	2 September 2010
Example Code	Replaced the Java example code with the new Java example code from the AWS SDK for Java. For more information, see the Java examples listed in <a href="#">Working with Amazon SQS (p. 13)</a> .	22 March 2010
Example Code	Replaced the C# and .NET example code with the new C# example code from the AWS SDK for .NET. For more information, see the C# examples listed in <a href="#">Working with Amazon SQS (p. 13)</a> .	11 November 2009
Update	Updated the guide to reflect changes in the sample code that AWS provides for Java, C#, Perl, PHP5, and VB.NET.	15 December 2008

# Glossary

---

## Glossary

---

Access Key ID	An identifier associated with your Secret Access Key. Used for request authentication.
AWS access key identifiers	Your Access Key ID and Secret Access Key. You receive these when you create an AWS account.
message ID	An identifier you get when you send a message to the queue.
Query	This is a type of HTTP request that generally uses only the GET or POST HTTP method and a query string with parameters.
queue URL	The URL uniquely identifying a queue.
receipt handle	An identifier you get when you receive a message from the queue. You must provide this identifier when deleting a message from the queue or when changing a message's visibility timeout.
Secret Access Key	A key that Amazon Web Services (AWS) assigns to you when you sign up for an AWS account. Used for request authentication.
visibility timeout	The length of time (in seconds) that a message that has been received from a queue will be invisible to other receiving components when they ask to receive messages. During the visibility timeout, the component that received the message usually processes the message and then deletes it from the queue.