# Amazon Simple Queue Service

## Getting Started Guide

## API Version 2012-11-05

# Amazon Simple Queue Service: Getting Started Guide

Copyright © 2015 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

# Table of Contents

# Welcome

**Topics**

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.

# Amazon SQS Resources

You may find the following related resources useful as you work with this service.

Amazon Simple Queue Service Developer Guide
> The developer guide provides a detailed discussion of the service. It includes an architectural overview and a programming reference.

Amazon Simple Queue Service API Reference
> 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

*Use Amazon SQS Queues to Determine When to Auto Scale*
> You can use Amazon SQS queues to help determine the load on an application, and when combined with Auto Scaling, you can scale the number of Amazon EC2 instances out or in depending upon the volume of traffic.

Amazon SQS Release Notes
> The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.

Product information for Amazon SQS
> The primary web page for information about Amazon SQS.

Amazon SQS Discussion Forums
> A community-based forum for developers to discuss technical questions related to Amazon SQS.

AWS Premium Support Center
> 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.

# Introduction to Amazon SQS

**Topics**

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 256 KB of text in any format. Any component can later retrieve the messages programmatically using the Amazon 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.

Amazon 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
  Amazon 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 262,144 bytes (256 KB) 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 SimpleDB 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.
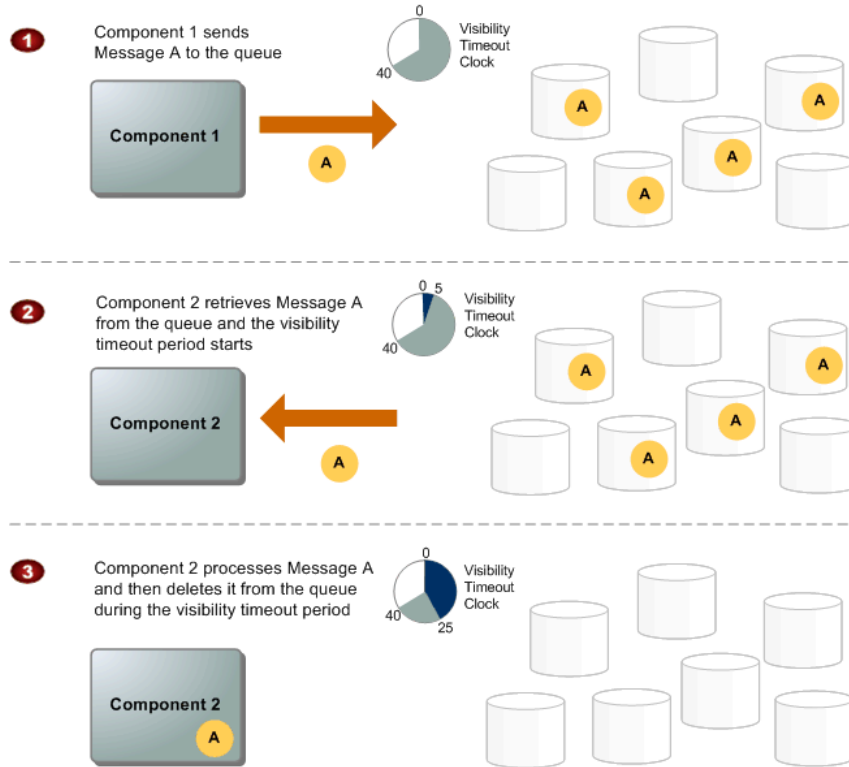
## Message Lifecycle

| 1 | Component 1 sends Message A to a queue, and the message is redundantly distributed across the SQS servers. |
|---|---|
| 2 | 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 *visibility timeout*. |
| 3 | 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 Amazon 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. 9).

For information about the other operations you can perform with Amazon SQS, see 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 |
|---|---|
| 1. Getting the Tools You Need (p. 8) | 1. Creating an AWS Account (p. 6) |
| | 2. Getting Your Access Key ID and Secret Access Key (p. 7) |
| | 3. Getting the Tools You Need (p. 8) |

## Creating an AWS Account

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.

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. Open http://aws.amazon.com/, and then click **Sign Up**.
2. Follow the on-screen instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

# Getting Your Access Key ID and Secret Access Key

**To get your access key ID and secret access key**

Access keys consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS. If you don't have access keys, you can create them by using the AWS Management Console. We recommend that you use IAM access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account.

> **Note**
> To create access keys, you must have permissions to perform the required IAM actions. For more information, see Granting IAM User Permission to Manage Password Policy and Credentials in *IAM User Guide.*

1. Open the IAM console.
2. From the navigation menu, click **Users**.
3. Select your IAM user name.
4. Click **User Actions**, and then click **Manage Access Keys**.
5. Click **Create Access Key**.

   Your keys will look something like this:

   - Access key ID example: AKIAIOSFODNN7EXAMPLE
   - Secret access key example: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

6. Click **Download Credentials**, and store the keys in a secure location.

   Your secret key will no longer be available through the AWS Management Console; you will have the only copy. Keep it confidential in order to protect your account, and never email it. 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 key.
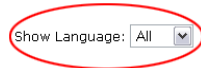
**Related topics**

- What Is IAM? in *IAM User Guide*
- AWS Security Credentials in *AWS General Reference*

# Getting the Tools You Need

If you want to use the sample code that goes with this guide, you must install the programming tools listed in this section.

> **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

## PHP 5

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

- PHP 5.2.1 or newer (for more information, go to http://php.net)
- AWS SDK for PHP (for more information, go to http://aws.amazon.com/sdkforphp/)

You must also:

- Add your credentials to your config.inc.php file, per the instructions in the section, "Configure the SDK Security Credentials," of the article, "Getting Started with the AWS SDK for PHP" http://aws.amazon.com/articles/4261
- Verify that your PHP environment passes the SDK Compatibility Test, `sdk_compatibility_test.php`, which is located in the subdirectory, `_compatibility_test`, of the downloaded files for the AWS SDK for PHP.
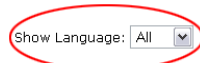
# Working with Amazon SQS

**Topics**

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 AWS Command Line Interface (CLI). You can also code with one of the following: Java, C#, 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.
>
> Show Language: All
>
> 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 the AWS Command Line Interface User Guide. You can also use **AWS Explorer** in the AWS Toolkit for Eclipse Getting Started Guide or in the AWS Toolkit for Visual Studio User Guide to become more familiar with Amazon SQS without writing code.

# AWS Management Console

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

### To access Amazon SQS with the AWS Management Console

- Sign in to the AWS Management Console and open the Amazon SQS console at https://console.aws.amazon.com/sqs/.

# 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. Unzip the `aws-java-sdk-<version>.zip` file to a directory designated as <sqs home> on your machine.

3. Specify your AWS credentials by following the instructions at Getting Started with the AWS SDK for Java.

4. Open the Amazon SQS sample at `<sqs home>/aws-java-sdk-<version>/samples/AmazonSimpleQueueService`.

# 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_<version>.msi`.

    The installation installs to `C:\Program Files (x86)\AWS SDK for .NET` by default.

3. Specify your AWS credentials by following the instructions at Configuring AWS Credentials.

4. Go to `C:\Program Files\AWS SDK for .NET\Samples\AmazonSQS_Sample` and open the solution:

    - `AmazonSQS_Sample.sln`

        **Note**
        Depending on your permissions, you might need to copy the sample to another directory.

5. Open the `App.config` file.

6. In the `AWSProfileName` setting, specify the name of the profile that you defined for your credentials.
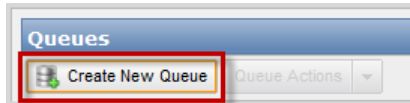
7. Save the file.

# 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`.
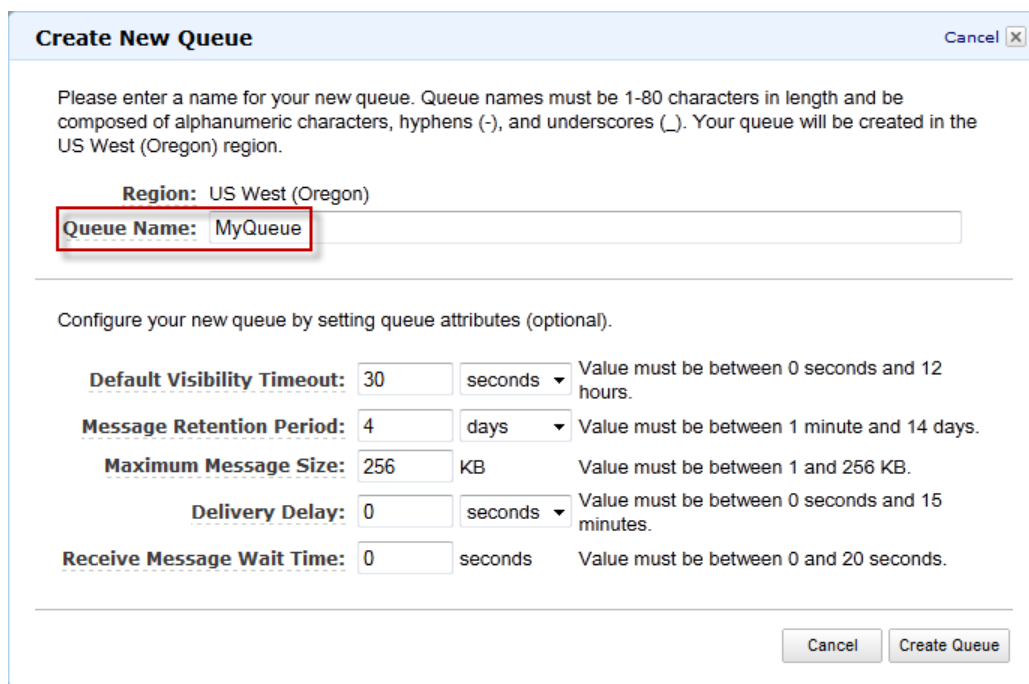
# AWS Management Console

**To run the sample**

1. Sign in to the AWS Management Console and open the Amazon SQS console at https://
   console.aws.amazon.com/sqs/.
2. Click **Create New Queue**.



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



4. Click **Create Queue**.
   Your new queue appears in the list of queues.
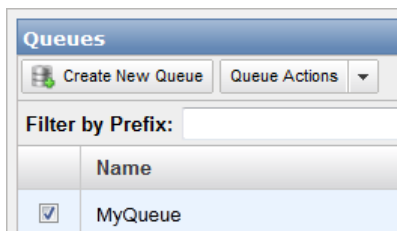
# Java

**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.QueueUrl;
```

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

# PHP 5

**To run the sample**

1. Run the sample, `SimpleQueueServiceSample.php`, on your web server by loading it in your browser.
   For example, if you are using an Apache Web Server, then you can place
   `SimpleQueueServiceSample.php` in the `htdocs` directory and then type
   **http://localhost/SimpleQueueServiceSample.php** in your browser.
2. The results will display on your browser window.

   The following section of the code creates a queue:

```
// Create the queue using the AWS-PHP-SDK function, create_queue, and store
it
```

```
// in the '$newqueue' variable.
$newqueue = $sqs->create_queue($queuename);

// Print the created queue results.
echo/ '<br></br><h2>CREATE QUEUE</h2>';
print '<pre>' . print_r($newqueue, true) . '<pre><br/><br/>';
```
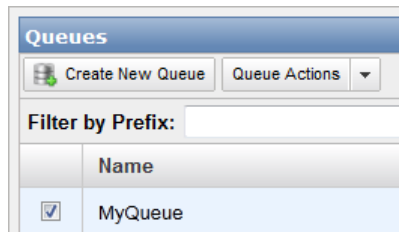
# Confirming the Queue Exists

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

## AWS Management Console

The AWS Management Console displays a list of your queues for the region you have selected.

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

- Select a region from the **Region** drop-down list, which is located on the top right of the console next to Help.
  The console displays all of your queues in that region.



## 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");
   foreach (String queueUrl in listQueuesResponse.QueueUrls)
   {
       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.

# 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**.
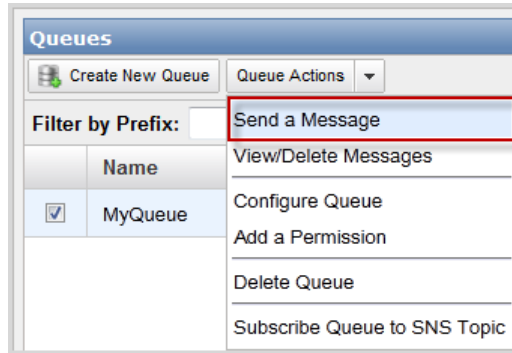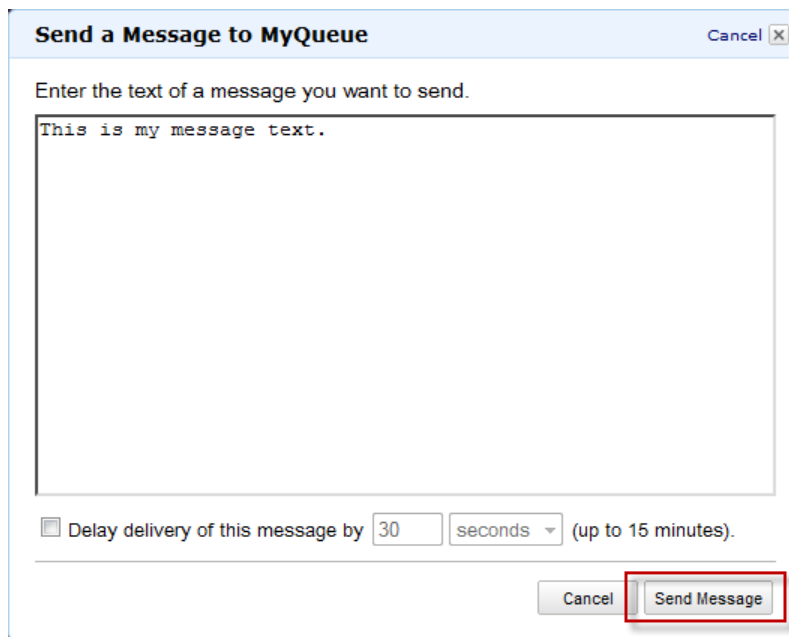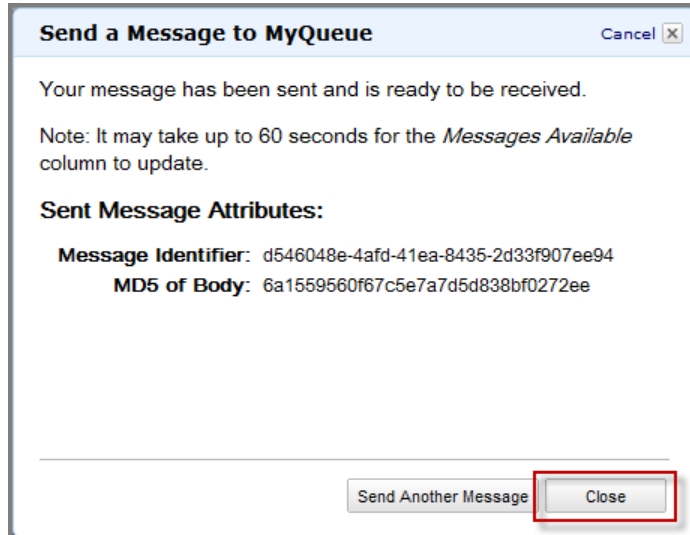


4.  In the **Send a Message to MyQueue** confirmation box click **Close**.

# 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

# PHP 5

**To run the sample**

1. Run the sample, `SimpleQueueServiceSample.php`, on your web server by loading it in your browser. For example, if you are using an Apache Web Server, then you can place `SimpleQueueServiceSample.php` in the `htdocs` directory and then type **http://localhost/SimpleQueueServiceSample.php** in your browser.

2. The results will display on your browser window.

   The following section of the code send a message to the queue:

```
// Send a message to the queue using the AWS SDK for PHP function, send_mes
sage.
$sndmessage = $sqs->send_message($url, 'HAVE A NICE DAY');

// The message body is an array element in the CFResponse Object [header].
$message = $sndmessage->header['x-aws-body'];

// Print the sent message.
echo '<h2>SEND MESSAGE</h2>';
print '<pre>' . print_r($message, true) . '<pre><br/><br/>';
```

# 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 Amazon SQS returns up to that maximum number. Because Amazon 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, in this example where the default of short polling is used, you should rerun the sample until you get the message.
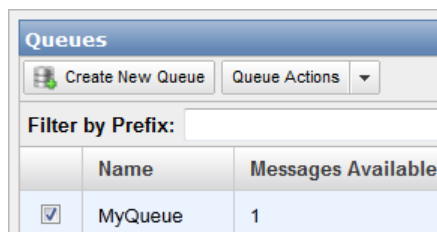
**Note**
When you design your own application determine whether short or long polling works better for your environment and application needs. For more information, see Amazon SQS Long Poll

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. 23).

# AWS Management Console
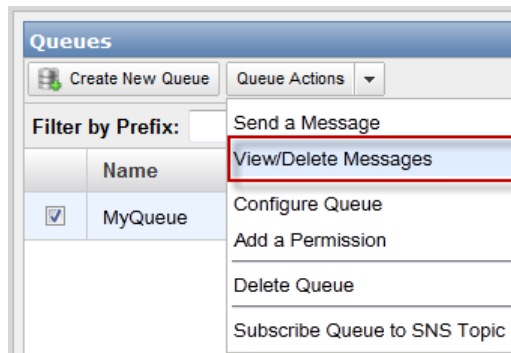
### To receive a message

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.
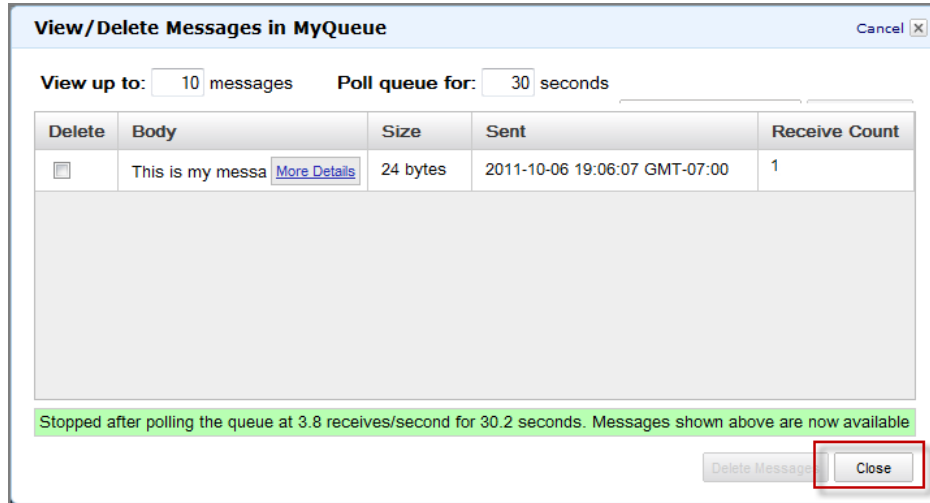
**View/Delete Messages in MyQueue**                              Cancel ☒

**View up to:** [10] messages    **Poll queue for:** [30] seconds

| Delete | Body | Size | Sent | Receive Count |
|--------|------|------|------|---------------|
| ☐ | This is my messa [More Details] | 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.

# 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:

```csharp
//Receiving a message
ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest();
receiveMessageRequest.QueueUrl = myQueueUrl;
ReceiveMessageResponse receiveMessageResponse = sqs.ReceiveMessage(receiveMes
sageRequest);

Console.WriteLine("Printing received message.\n");
foreach (Message message in receiveMessageResponse.Messages)
{
    Console.WriteLine("  Message");
    Console.WriteLine("    MessageId: {0}", message.MessageId);
    Console.WriteLine("    ReceiptHandle: {0}", message.ReceiptHandle);
    Console.WriteLine("    MD5OfBody: {0}", message.MD5OfBody);
    Console.WriteLine("    Body: {0}", message.Body);

    foreach (KeyValuePair<string, string> entry in message.Attributes)
    {
        Console.WriteLine("  Attribute");
        Console.WriteLine("    Name: {0}", entry.Key);
        Console.WriteLine("    Value: {0}", entry.Value);
    }
}
String messageRecieptHandle = receiveMessageResponse.Messages[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.

# PHP 5

**To run the sample**

1. Run the sample, `SimpleQueueServiceSample.php`, on your web server by loading it in your browser. For example, if you are using an Apache Web Server, then you can place `SimpleQueueServiceSample.php` in the `htdocs` directory and then type **http://localhost/SimpleQueueServiceSample.php** in your browser.

2. The results will display on your browser window.

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

```
// Receive a message from the queue using the AWS SDK for PHP function, re
ceive_message.
$received = $sqs->receive_message($url);

// Print the received message results.
echo '<h2>RECEIVE MESSAGE</h2>';
print '<pre>' . print_r($received, true) . '<pre><br/><br/>';

// Store the receipt handle, which is a CFSimpleXML object filtered from the

// CFSimpleXML [body].
$receipthandle = $received->body->ReceiveMessageResult->Message->ReceiptHandle;

// Print receipt handle.
echo '<h2>ReceiptHandle</h2>';
echo '<pre>' . print_r($receipthandle, true) . '<pre><br/><br/>';
```

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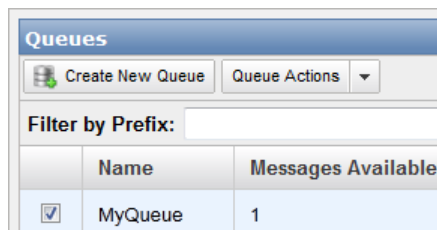
> **Note**
> If you don't have the receipt handle for the message, you can call `ReceiveMessage` again and receive the message again. Each time you receive the message, you get a different receipt handle. Use the latest receipt handle when calling `DeleteMessage`; otherwise, your message might not be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

The following examples demonstrate how to delete the message from your `MyQueue` queue.
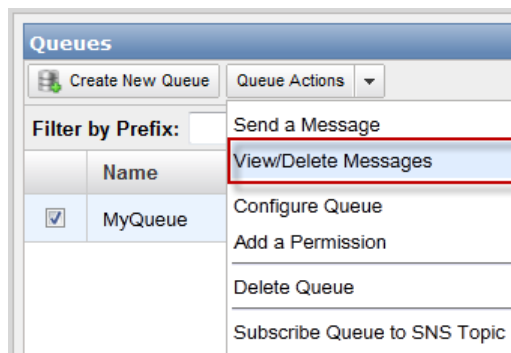
## AWS Management Console

**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.
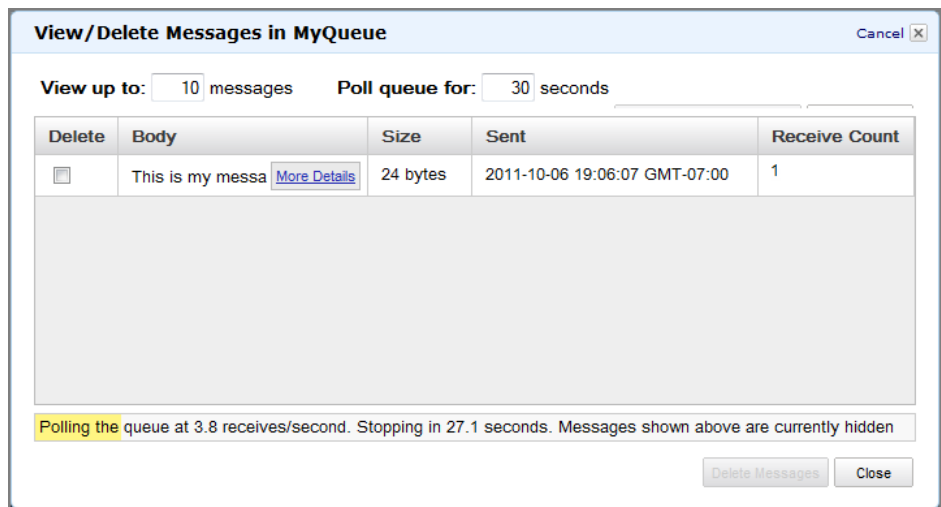
- Messages will come from the front of the queue unless other applications are also reading from the queue.

- 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.

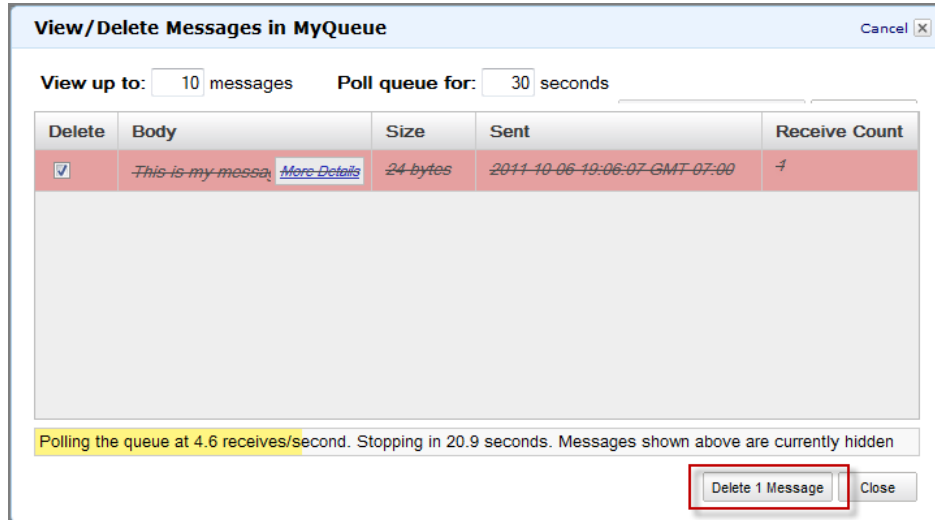The **View/Delete Messages** dialog box displays a message from the queue.



4.   Select the message you want to delete.

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



6. Click **Yes, Delete Checked Messages**.
   The selected message is deleted.

7. Click **Close** to close the **View/Delete Messages** dialog box.

# 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.

# PHP 5

**To run the sample**

1. Run the sample, `SimpleQueueServiceSample.php`, on your web server by loading it in your browser. For example, if you are using an Apache Web Server, then you can place `SimpleQueueServiceSample.php` in the `htdocs` directory and then type **http://localhost/SimpleQueueServiceSample.php** in your browser.
2. The results will display on your browser window.

   The following section of the code deletes a message from the queue:

```
// Deletes a message from a specific queue using the receipt handle
$deletemessage = $sqs->delete_message($url, $receipthandle);

// Print the delete message results.
echo '<h2>DELETE MESSAGE</h2>';
echo '<pre>' . print_r($deletemessage, true) . '<pre>';
```

# You've 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. 27).

# 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 Amazon SQS content available in Articles & Tutorials. Specifically, two webcasts are 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. 9)). You can look at other Amazon SQS 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
- Python
- 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, see the product description at http://aws.amazon.com/iam or the technical documentation at IAM User Guide.

# Document History

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

- **API version:** 2012-11-05
- **Latest documentation update:** November 21, 2012

| Change | Description | Date Changed |
|---|---|---|
| New feature | Amazon SQS is now available through the AWS Management Console. For an example of how to create a queue, send a message, and receive a message with the AWS Management Console, see Working with Amazon SQS (p. 9). | November 21, 2012 |
| New feature | This service now integrates with AWS Identity and Access Management (IAM). For more information, go to AWS Services that Support IAM in IAM User Guide. | September 2, 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 Working with Amazon SQS (p. 9). | March 22, 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 Working with Amazon SQS (p. 9). | November 11, 2009 |
| Update | Updated the guide to reflect changes in the sample code that AWS provides for Java, C#, Perl, PHP5, and VB.NET. | December 15, 2008 |