**Introduction to Amazon SQS**

**Topics**

* [Overview of Amazon SQS](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/Introduction.html#Overview)
* [Features](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/Introduction.html#SQSFeatures)
* [Message Lifecycle](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/Introduction.html#MessageLifecycle)
* [About the Samples](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/Introduction.html#about-the-samples)

## 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. Messages larger than 256 KB can be managed using the Amazon SQS Extended Client Library for Java, which uses Amazon S3 for storing larger payloads.

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.

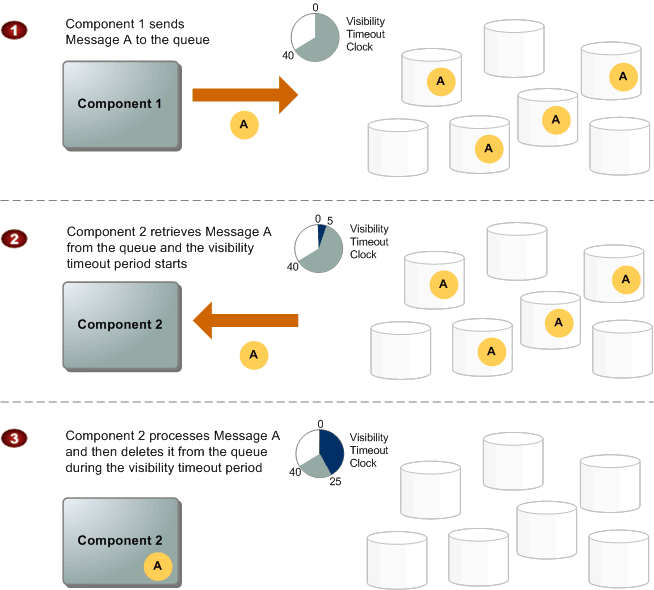
* **Variable message size** – Your messages can be up to 262,144 bytes (256 KB) in size

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 object. For more information, see [Managing Amazon SQS Messages with Amazon S3](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/s3-messages.html). Alternatively, you can split the larger message into smaller ones.

* **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, calledMessage 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 thevisibility 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](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_SetQueueAttributes.html" \t "_blank).

## About the Samples

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

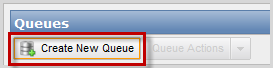
# Creating a Queue

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

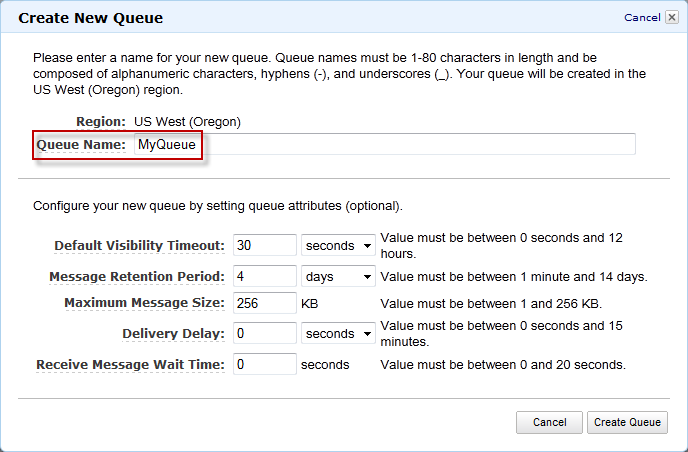
## AWS Management Console

**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**.

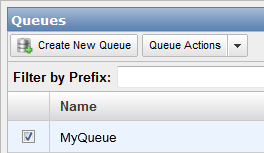


1. In the **Create New Queue** dialog box, enter MyQueue in the **Queue Name** field, and leave the default value settings for the remaining fields.



1. 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 CreateQueueRequest().withQueueName("MyQueue");

String myQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();

1. Compile and 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 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.

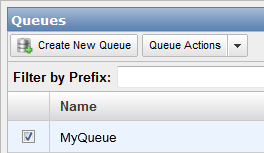
## AWS Management Console

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();

1. 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.

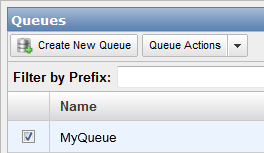
# Sending a Message

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 messageThis is my message text. to your MyQueue queue.

## AWS Management Console

**To send a message**

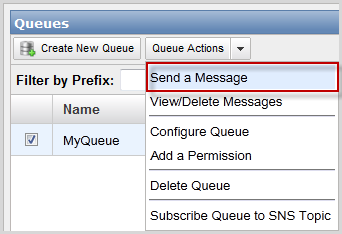
1. In the AWS Management Console select a queue.



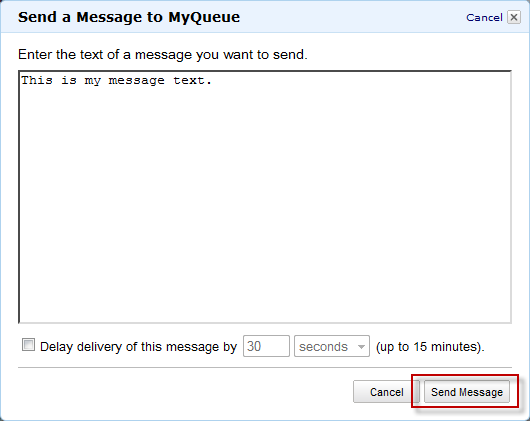
1. Select **Send a Message** from the **Queue Actions** drop-down list.

**Note**

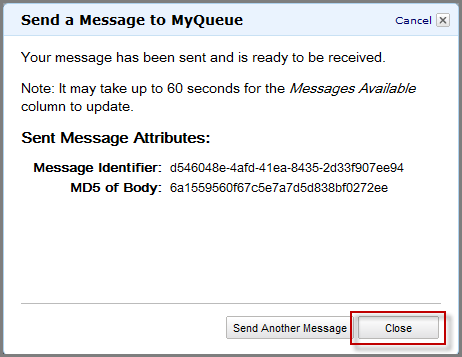
The **Queue Actions** drop-down list is available only if a queue is selected.



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



1. 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."));

1. 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

# Receiving a Message

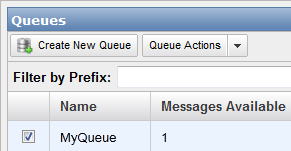
Now that a message is in the queue, you can receive it (retrieve it from the queue). When requesting to get a message from the queue, you can't specify which message to get. You simply specify the maximum number of messages you want to get (up to 10), and Amazon SQS returns up to that maximum number. 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.

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.

## AWS Management Console

**To receive a message**

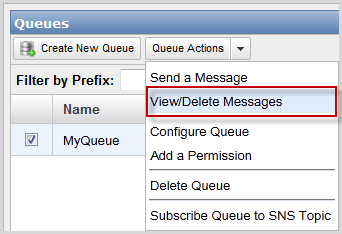
1. In the AWS Management Console, select a queue.



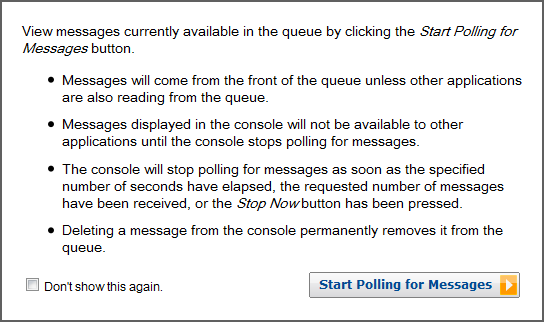
1. Select **View/Delete Messages** from the **Queue Actions** drop-down list.

**Note**

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



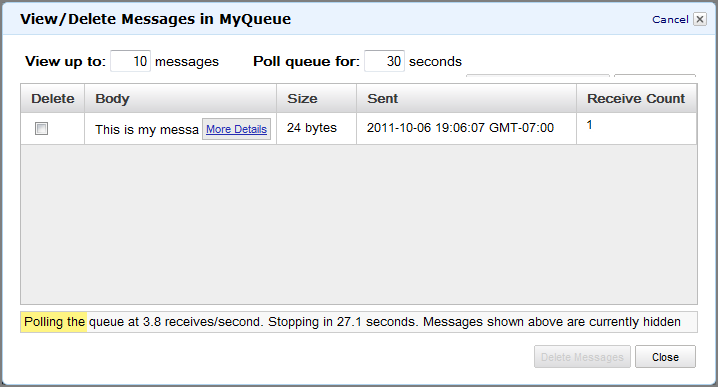
1. Click **Start Polling for Messages** to receive a message from the queue.



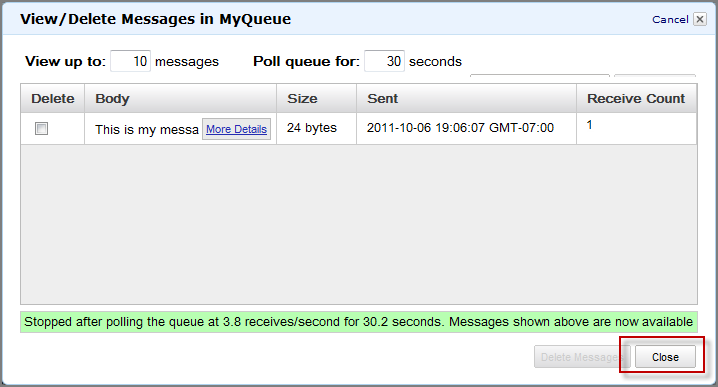
**Note**

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.



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.



1. 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 ReceiveMessageRequest(myQueueUrl);

List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages();

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();

1. 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.

# 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 toDeleteQueue, even if the queue has messages in it.

**Note**

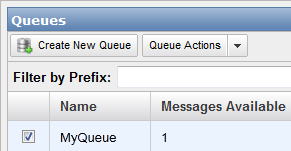
If you don't have the receipt handle for the message, you can callReceiveMessage again and receive the message again. Each time you receive the message, you get a different receipt handle. Use the latest receipt handle when calling DeleteMessage; otherwise, your message might not be deleted from the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

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

## AWS Management Console

**To delete a message**

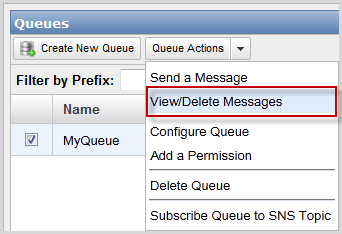
1. In the AWS Management Console, select a queue.



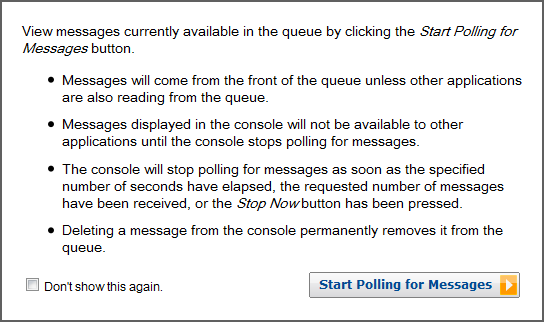
1. Select **View/Delete Messages** from the **Queue Actions** drop-down list.

**Note**

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



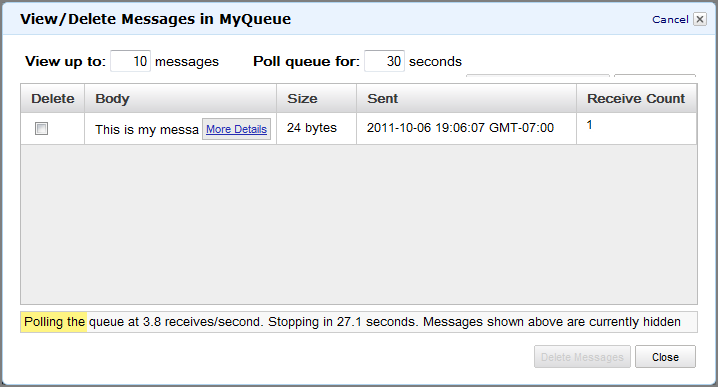
1. Click **Start Polling for Messages** to view a message from the queue.



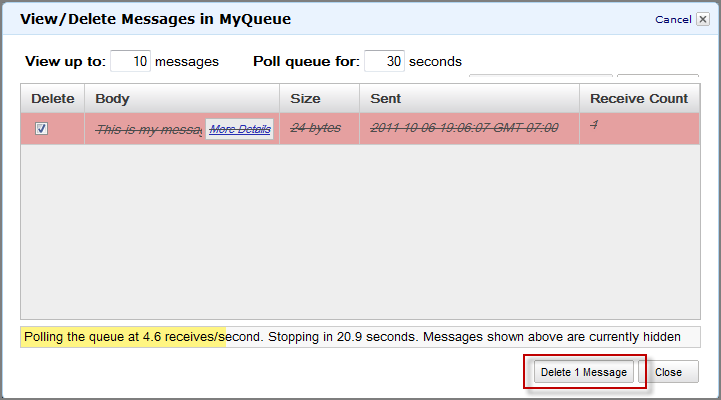
**Note**

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.

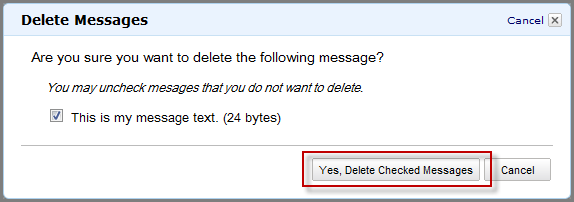


1. Select the message you want to delete.



1. Click **Delete 1 Message** to delete the selected message.

A **Delete Messages** confirmation dialog box appears.



1. Click **Yes, Delete Checked Messages**.

The selected message is deleted.

1. 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));

1. 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.

# Visibility Timeout

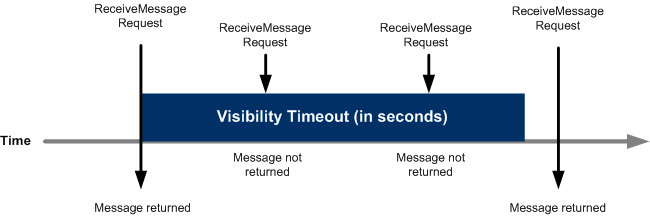
**Topics**

* [General Recommendations for Visibility Timeout](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/AboutVT.html#AboutVT-general-recommendations-for-visibility-timeout)
* [Extending a Message's Visibility Timeout](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/AboutVT.html#AboutVT-extending-message-visibility-timeout)
* [Terminating a Message's Visibility Timeout](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/AboutVT.html#AboutVT-terminating-message-visibility-timeout)
* [API Actions Related to Visibility Timeout](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/AboutVT.html#visAPI)

When a consuming component in your system receives and processes a message from the queue, the message remains in the queue. Why doesn't Amazon SQS automatically delete it?

Because your system is distributed, there's no guarantee that the component will actually receive the message (it's possible the connection could break or the component could fail before receiving the message). Therefore, Amazon SQS does not delete the message, and instead, your consuming component must delete the message from the queue after receiving and processing it.

Immediately after the component receives the message, the message is still in the queue. However, you don't want other components in the system receiving and processing the message again. Therefore, Amazon SQS blocks them with a visibility timeout, which is a period of time during which Amazon SQS prevents other consuming components from receiving and processing that message. The following figure and discussion illustrate the concept.



**Note**

There is a 120,000 limit for the number of inflight messages per queue. Messages are inflight after they have been received from the queue by a consuming component, but have not yet been deleted from the queue. If you reach the 120,000 limit, you will receive an OverLimit error message from Amazon SQS. To help avoid reaching the limit, you should delete the messages from the queue after they have been processed. You can also increase the number of queues you use to process the messages.

## General Recommendations for Visibility Timeout

The visibility timeout clock starts ticking once Amazon SQS returns the message. During that time, the component processes and deletes the message. But what happens if the component fails before deleting the message? If your system doesn't call [DeleteMessage](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_DeleteMessage.html" \t "_blank) for that message before the visibility timeout expires, the message again becomes visible to the[ReceiveMessage](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_ReceiveMessage.html) calls placed by the components in your system and it will be received again. If a message should only be received once, your system should delete it within the duration of the visibility timeout.

Each queue starts with a default setting of 30 seconds for the visibility timeout. You can change that setting for the entire queue. Typically, you'll set the visibility timeout to the average time it takes to process and delete a message from the queue. When receiving messages, you can also set a special visibility timeout for the returned messages without changing the overall queue timeout.

We recommend that if you have a system that produces messages that require varying amounts of time to process and delete, you create multiple queues, each with a different visibility timeout setting. Your system can then send all messages to a single queue that forwards each message to another queue with the appropriate visibility timeout based on the expected processing and deletion time for that message.

## Extending a Message's Visibility Timeout

When you receive a message from a queue and begin processing it, you may find the visibility timeout for the queue is insufficient to fully process and delete that message. To give yourself more time to process the message, you can extend its visibility timeout by using the [ChangeMessageVisibility](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_ChangeMessageVisibility.html" \t "_blank) action to specify a new timeout value. Amazon SQS restarts the timeout period using the new value.

For example, let's say the timeout for the queue is 30 seconds, and you receive a message from that queue. When you're 20 seconds into the timeout for that message (i.e., you have 10 seconds left), you want to give yourself 60 more seconds, so you immediately callChangeMessageVisibility for the message with *VisibilityTimeout* set to 60 seconds. This means that you extended the message's visibility timeout from 30 seconds to 80 seconds: 20 seconds from the initial timeout setting plus 60 seconds from when you changed the timeout.

When you extend a message's visibility timeout, the new timeout applies only to that particular receipt of the message. ChangeMessageVisibility does not affect the timeout for the queue or later receipts of the message. If for some reason you don't delete the message and receive it again, its visibility timeout is the original value set for the queue.

## Terminating a Message's Visibility Timeout

When you receive a message from the queue, you might find that you actually don't want to process and delete that message. Amazon SQS allows you to terminate the visibility timeout for a specific message, which immediately makes the message visible to other components in the system to process. To do this, you call [ChangeMessageVisibility](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_ChangeMessageVisibility.html" \t "_blank) with*VisibilityTimeout*=0 seconds.

## API Actions Related to Visibility Timeout

The following table lists the API actions to use to manipulate the visibility timeout. Use each action's *VisibilityTimeout* parameter to set or get the value.

| **To do this...** | **Use this action** |
| --- | --- |
| Set the visibility timeout for a queue | [SetQueueAttributes](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_SetQueueAttributes.html) |
| Get the visibility timeout for a queue | [GetQueueAttributes](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_GetQueueAttributes.html) |
| Set the visibility timeout for the received messages without affecting the queue's visibility timeout | [ReceiveMessage](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_ReceiveMessage.html) |
| Extending or terminating a message's visibility timeout | [ChangeMessageVisibility](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_ChangeMessageVisibility.html) |
| Extending or terminating the visibility timeout for up to ten messages. | [ChangeMessageVisibilityBatch](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_ChangeMessageVisibilityBatch.html) |

# Amazon SQS FAQs

## Introduction and Getting Started

Q: What is Amazon SQS?

Amazon Simple Queue Service (Amazon SQS) offers a reliable, highly scalable hosted queue for storing messages as they travel between computers. By using Amazon SQS, developers can simply move data between distributed application components performing different tasks, without losing messages or requiring each component to be always available. Amazon SQS makes it easy to build an automated workflow, working in close conjunction with the Amazon Elastic Compute Cloud (Amazon EC2) and the other AWS infrastructure web services.

#### Manage Your AWS Resources

[Sign in to the Console](https://console.aws.amazon.com/console/home)

Q: What can I do with Amazon SQS?

Amazon SQS is a web service that gives you access to a message queue that can be used to store messages while waiting for a computer to process them. This allows you to quickly build message queuing applications that can be run on any computer on the internet. Since Amazon SQS is highly scalable and you only pay for what you use, you can start small and grow your application as you wish, with no compromise on performance or reliability. This lets you focus on building sophisticated message-based applications, without worrying about how the messages are stored and managed. You can use Amazon SQS with software applications in various ways. For example, you can:

* Integrate Amazon SQS with other AWS infrastructure web services to make applications more reliable and flexible.
* Use an Amazon SQS queue as a queue of work where each message is a task that needs to be completed by a process. One or many computers can read tasks from the queue and perform them.
* Have Amazon SQS help a browser-based application receive notifications from a server. The application server can add the notifications to a queue, which the browser can poll—even if there is a firewall between them.
* Keep notifications of significant events in a business process in an Amazon SQS queue. Each event can have a corresponding message in a queue, and applications that need to be aware of the event can read and process the messages.

Q: How can I get started using Amazon SQS?

To sign up for Amazon SQS, click the “Sign up for This Web Service” button on the Amazon SQS detail page. You must have an Amazon Web Services account to access this service; if you do not already have one, you will be prompted to create one when you begin the Amazon SQS sign-up process. After signing up, please visit the [AWS Management Console](http://aws.amazon.com/console/) or refer to the Amazon SQS documentation and sample code in the Resource Center to begin using Amazon SQS.

Q: What are the benefits of Amazon SQS vs. homegrown or packaged message queuing systems?

Using Amazon SQS provides several advantages over building your own software for managing message queues or using a commercial or open-source message queuing system. The alternatives require significant upfront time to develop and configure compared to integrating with an existing web service. Also, these alternatives require ongoing hardware and system administrative resources to operate. The complexity of configuring and managing these systems is compounded if they need to be configured to provide redundant storage of messages so messages are not lost if hardware fails. By contrast, Amazon SQS requires no administrative overhead and little configuration.

Q: Does Amazon use Amazon SQS for its own applications?

Yes. Developers within Amazon use Amazon SQS for a wide variety of projects and represent a large number of Amazon SQS messages per day. Applications using Amazon SQS include key business processes for the Amazon.com retail web site and Amazon Web Services.

## Billing

Q: What can I do with the Amazon SQS free tier?

The Amazon SQS free tier provides 1 million requests per month at no charge. Many small scale applications may be able to operate entirely within this free tier limit. Data transfer charges still apply. The free tier is a monthly offer. Free usage does not accumulate across months.

Q: How much does Amazon SQS cost?

You pay only for what you consume, and there is no minimum fee. You pay $0.50 for every 1 million requests, plus data transfer charges for data transferred out of Amazon SQS. Data transferred between Amazon SQS and Amazon EC2 in the U.S. is free of charge. Please see the [SQS pricing page](https://aws.amazon.com/sqs/pricing/) for more detail.

Q: Do Amazon SQS batch operations cost more than other requests?

Batch operations which include; SendMessageBatch, DeleteMessageBatch and ChangeMessageVisbilityBatch all cost the same as other Amazon SQS requests.

Q: How will I be charged and billed for my use of Amazon SQS?

There are no set-up fees to begin using the service. At the end of the month, your credit card will automatically be charged for that month’s usage. You can view your charges for the current billing period at any time on the Amazon Web Services web site by logging into your Amazon Web Services account and clicking “Account Activity” under “Your Web Services Account”.

Q: Do your prices include taxes?

Except as otherwise noted, our prices are exclusive of applicable taxes and duties, including VAT and applicable sales tax. For customers with a Japanese billing address, use of the Asia Pacific (Tokyo) Region is subject to Japanese Consumption Tax. [Learn more](https://aws.amazon.com/c-tax-faqs/).

## Features, Functionality and Interfaces

Q: Can Amazon SQS be used with other AWS services?

Amazon SQS can be used with Amazon EC2, as well as Amazon S3 and Amazon DynamoDB, to make applications more flexible and scalable. A common use case is to create an integrated and automated workflow, where multiple components or modules need to communicate with each other, but can’t all process the same amount of work simultaneously. In this case, Amazon SQS queues carry messages to be processed in an orderly fashion by the user’s application running on Amazon EC2 instances. The Amazon EC2 instances can read the queue, process the job, and then post the results as messages to another Amazon SQS queue (possibly for further processing by another application). Because Amazon EC2 allows applications to scale up and down dynamically, application developers can easily vary the number of compute instances based on the amount of work in the SQS queues, to ensure that jobs are executed in a timely manner.

For example, here is how a video transcoding website uses Amazon EC2, Amazon SQS, Amazon S3, and Amazon DynamoDB together. End users submit videos to be transcoded to the website. The videos are stored in Amazon S3, and a message (“the request message”) is placed in an Amazon SQS queue (“the incoming queue”) with a pointer to the video and to the target video format in the message. The transcoding engine, running on a set of Amazon EC2 instances, reads the request message from the incoming queue, retrieves the video from Amazon S3 using the pointer, and transcodes the video into the target format. The converted video is put back into Amazon S3 and another message (“the response message”) is placed in another Amazon SQS queue (“the outgoing queue”) with a pointer to the converted video. At the same time, metadata about the video (e.g. format, date created and length) can be indexed into Amazon DynamoDB for easy querying. During this whole workflow, a dedicated Amazon EC2 instance can constantly monitor the incoming queue and, based on the number of messages in the incoming queue, is able to dynamically adjust the number of transcoding Amazon EC2 instances to meet customers’ response time requirements.

Q: How do I interface with Amazon SQS?

Amazon SQS provides simple APIs that are designed to work with any Internet-development toolkit. The operations are intentionally made simple to work with messages and queues.

Q: What are the available operations for queues? Who can perform operations on a queue?

The following set of operations is defined for Amazon SQS: CreateQueue, ListQueues, DeleteQueue, SendMessage, SendMessageBatch, ReceiveMessage, DeleteMessage, DeleteMessageBatch, PurgeQueue, ChangeMessageVisibility, ChangeMessageVisibilityBatch, SetQueueAttributes, GetQueueAttributes, GetQueueUrl, AddPermission and RemovePermission. Operations can be performed only by the AWS account owner or an AWS account that the account owner has delegated to.

Q: How are messages identified in the system?

All messages have a globally unique ID that Amazon SQS returns when the message is delivered to the queue. The ID isn’t required in order to perform any further actions on the message, but it’s useful for tracking whether a particular message in the queue has been received. When you receive a message from the queue, the response includes a receipt handle, which you must provide when deleting the message.

Q: Does Amazon SQS offer Dead Letter Queues (DLQs) ?

Yes. A DLQ is an SQS queue which you configure to receive messages from other SQS queues - referred to as “source queues.” Typically, you set up a DLQ to receive messages after a max number of processing atttempts have been reached. DLQ provides the ability to isolate messages that could not be processed for analysis out of band.

A DLQ is just like any other SQS queue. Messages can be sent to it and received from it like any other SQS queues. You can create a DLQ from the SQS API and the SQS console.

You can learn about how to create and use Dead Letter Queues from the [SQS Developer Guide](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/SQSDeadLetterQueue.html).

Q: Does Amazon SQS provide first-in-first-out (FIFO) access to messages?

No, Amazon SQS does not guarantee FIFO access to messages in Amazon SQS queues, mainly because of the distributed nature of the Amazon SQS. If you require specific message ordering, you should design your application to handle it.

Q: What is the maximum limit for message visibility?

Amazon SQS supports up to 12 hours maximum visibility timeout.

Q: Does SQS support message metadata?

Yes. Amazon SQS allows you to send up to 10 attributes on each message. With message attributes, you can separate the body of a message from the metadata that describes it. This allows you to route, process and store information with greater speed and intelligence because your applications no longer have to inspect an entire message before understanding what processing steps are necessary.

SQS message attributes take the form of name-type-value triples. Types supported include: String, Binary and Number (including integers, floating point, and doubles).

Please refer to the [SQS Documentation](https://aws.amazon.com/documentation/sqs/) for additional information.

Q: How can a developer determine “time-in-queue”?

Developers can request the “SentTimestamp” attribute when receiving a message. Subtracting that value from the current time results in the “time-in-queue” value.

Q: What is the “SenderId” attribute value of a message in the case of anonymous access?

Amazon SQS provides the IP address when the AWS account ID is not available such as when an anonymous user sends a message.

Q: What is SQS Long Polling?

SQS long polling is a way to retrieve messages from your SQS queues. While the regular SQS short polling returns immediately, even if the queue being polled is empty, SQS long polling doesn’t return a response until a message arrives in the queue, or the long poll times out. SQS long polling makes it easy and inexpensive to retrieve messages from your SQS queue as soon as they are available.

Q: Is there any additional charge for using SQS long polling?

No, long polling ReceiveMessage API calls are billed exactly the same as short polling ReceiveMessage API calls.

Q: When should I use SQS long polling, and when should I use SQS short polling?

In almost all cases, SQS long polling is preferable to SQS short polling. Long polling requests allow your queue consumers to receive messages as soon as they arrive in your queue, while reducing the number of empty ReceiveMessageResponses you encounter. Combined together, SQS long polling results in higher performance at reduced cost for the majority of use cases. However, if your application is written to expect an immediate response from a ReceiveMessage call, you may not be able to take advantage of long polling without some application modifications. For example, if your application has a single thread polling multiple queues, switching from short polling to long polling will likely not work, as the single thread will wait for the long poll timeout on any empty queues, delaying the processing of any queues which may contain messages. In such an application, it is recommended that a single thread be used to process only one queue, allowing for the application to take advantage of the benefits SQS long polling provides.

Q: What value should I use for my long polling timeout?

In general, the maximum long poll timeout of 20 seconds should be used. Higher long poll timeout values will reduce the number of empty ReceiveMessageResponses returned, so try to set your long poll timeout as high as possible. If the 20 second maximum does not work for your application, you can choose a shorter long poll timeout, down to as low as 1 second. All of the AWS SDK’s work with 20 second long polls by default. If you are not using an AWS SDK to access SQS, or if you’ve specially configured your AWS SDK to have a shorter timeout, you may need to modify your SQS client to allow for longer requests or use a shorter long poll timeout.

Q: What is the AmazonSQSBufferedAsync client ?

The AmazonSQSBufferedAsync client provides a new implementation for the AmazonSQSAsync client interface that adds several important features. First, the AmazonSQSBufferedAsync client supports automatic batching of multiple SendMessage, DeleteMessage or ChangeMessageVisibility requests into batches of each type, without any changes required by the application. Second, the AmazonSQSBufferedAsync client supports prefetching of messages into a local buffer, which allows your application to immediately process messages from SQS without waiting for them to be retrieved from SQS. Combined together, automatic batching and prefetching increases the throughput and reduces the latency of your application, while saving you money by making fewer SQS requests.

Q: How do I download the AmazonSQSBufferedAsync client?

You can download the AmazonSQSBufferedAsync client as part of the most recent AWS SDK for java, available at<http://aws.amazon.com/sdkforjava/>.

Q: Which languages does the AmazonSQSBufferedAsyncClient support?

At this time, the AmazonSQSBufferedAsyncClient is only supported for Java. We will be releasing the AmazonSQSBufferedAsyncClient in other languages in the near future.

Q: Do I have to rewrite my application to use the AmazonSQSBufferedAsync client?

No, the AmazonSQSBufferedAsync client is implemented as a drop in replacement for the existing AmazonSQSAsync client. You can update your application to use the latest AWS SDK, change your client construction to use the AmazonSQSBufferedAsync client instead of the AmazonSQSAsync client, and your application will receive the added benefits of automatic batching and prefetching.

Q: How can I subscribe SQS queues to receive notifications from SNS topics?

Select the SQS queue in the SQS console, and from the ‘Queue Actions’ in the menu bar, select ‘Subscribe Queue to SNS Topic’ from the drop-down list. In the subscribe dialog box, select the topic from the ‘Choose a Topic’ drop-down list, and click the ‘Subscribe’ button. For complete step-by-step instructions, please refer to the [Amazon SQS documentation](https://aws.amazon.com/documentation/sqs/).

Q: How can I fanout identical messages to multiple SQS queues?

Create an SNS topic first using SNS. Then create and subscribe multiple SQS queues to the SNS topic. Now whenever a message is sent to the SNS topic, the message will be fanned out to the SQS queues, i.e. SNS will deliver the message to all the SQS queues that are subscribed to the topic.

Q: Can I delete all the messages in a queue without deleting the queue itself?

Yes, you can delete all the messages in an SQS queue using the PurgeQueue action. When you purge a queue, all the messages previously sent to the queue will be deleted. Since your queue and its attributes will remain, there is no need to reconfigure the queue to continue using it. If you only need to delete specific messages, you can use the DeleteMessage or DeleteMessageBatch actions.

## Security and Reliability

Q: How reliably is my data stored in Amazon SQS?

Amazon SQS stores all queue and message information in Amazon’s network of highly reliable, highly available data centers. All messages are stored redundantly on multiple servers and in multiple data centers, which means that no single computer or network failure renders SQS messages inaccessible.

Q: How can I secure the messages in my queues?

 Authentication mechanisms are provided to ensure that messages stored in Amazon SQS queues are secured against unauthorized access. Only the AWS account owners can access the queues they create.

Amazon SQS uses proven cryptographic methods to authenticate your identity, either through the use of your Access Key ID and request signature, or through the use of an X.509 certificate. For the details of how to use either of these authentication mechanisms with Amazon SQS, please see the Amazon SQS Developer Guide.

For additional security, you could build your application to encrypt messages before they are placed in a queue.

Q: How does Amazon SQS allow multiple readers to access the same message queue, without losing messages or processing them many times?

Every Amazon SQS queue has a configurable visibility timeout. For the designated amount of time after a message is read from a queue, it will not be visible to any other reader. As long as the amount of time that it takes to process the message is less than the visibility timeout, every message will be processed and deleted. In the event that the component processing the message fails or becomes unavailable, the message will again become visible to any component reading the queue once the visibility timeout ends. This allows you to have many components all reading messages from the same queue, with each working to process different messages.

Q: How many times will I receive each message?

Amazon SQS is engineered to provide “at least once” delivery of all messages in its queues. Although most of the time each message will be delivered to your application exactly once, you should design your system so that processing a message more than once does not create any errors or inconsistencies.

Q: Why are there separate ReceiveMessage and DeleteMessage operations?

When Amazon SQS returns a message to you, that message stays in the queue, whether or not you actually received the message. You are responsible for deleting the message; the delete request acknowledges that you’re done processing the message. If you don’t delete the message, Amazon SQS will deliver it again on another receive request.

Q: Can a deleted message be received again?

Yes, under rare circumstances you might receive a previously deleted message again. This can occur in the rare situation in which a DeleteMessage operation doesn’t delete all copies of a message because one of the servers in the distributed Amazon SQS system isn’t available at the time of the deletion. That message copy can then be delivered again. You should design your application so that no errors or inconsistencies occur if you receive a deleted message again.

Q: What happens if I issue a DeleteMessage request on a previously deleted message?

SQS returns a “success” response.

## Limits, Restrictions

Q: How long can I keep my messages in Amazon SQS queues?

The SQS message retention period is configurable and can be set anywhere from 1 minute to 2 weeks. The default is 4 days and once the message retention limit is reached your messages will be automatically deleted. The option for longer message retention provides greater flexibility to allow for longer intervals between message production and consumption.

Q: How do I configure SQS to support longer message retention?

To configure the message retention period, set the MessageRetentionPeriod attribute using the SetQueueAttributes method. This attribute is used to specify the number of seconds a message will be retained by SQS. Currently the default value for the message retention period is 4 days. Using the MessageRetentionPeriod attribute, the message retention period can be set anywhere from 60 seconds (1 minute), up to 1209600 seconds (14 days). Please consult the [Amazon SQS API Reference Guide](https://aws.amazon.com/documentation/sqs/) for more detail about how to work with this new message attributes.

Q: How do I configure the maximum message size for SQS?

To configure the maximum message size, set the MaximumMessageSize attribute using the SetQueueAttributes method. This attribute specifies the limit on how many bytes an SQS message can contain. It can be set anywhere from 1024 bytes (1KB), up to 262144 bytes (256KB). Please consult the [Amazon SQS API Reference Guide](https://aws.amazon.com/documentation/sqs/) for more detail about how to work with this message attributes.

To send messages larger than 256KB, you can use the [Amazon SQS Extended Client Library for Java](https://github.com/awslabs/amazon-sqs-java-extended-client-lib).

Q: What kind of data can go in a message?

Amazon SQS messages can contain up to 256KB of text data, including XML, JSON and unformatted text. The following Unicode characters are accepted:

#x9 | #xA | #xD | [#x20 to #xD7FF] | [#xE000 to #xFFFD] | [#x10000 to #x10FFFF]

(according to http://www.w3.org/TR/REC-xml/#charsets).

Q: How big can Amazon SQS queues be?

A single queue may contain an unlimited number of messages, and you can create an unlimited number of queues.

Q: Are there restrictions on the names of Amazon SQS queues?

Is there a size limit on the name of Amazon SQS queues? Can a queue name be reused? Queue names are limited to 80 characters. Alphanumeric characters plus hyphens (-) and underscores (\_) are allowed. Queue names must be unique within an AWS account. After you delete a queue, you can reuse the queue name.

Q: What happens if there is no activity against a queue for an extended period of time?

We reserve the right to delete a queue if none of the following requests have been issued against the queue for more than 30 consecutive days: SendMessage, ReceiveMessage, DeleteMessage, GetQueueAttributes and SetQueueAttributes. You should design your application with this in mind. Note that queues that are used as Dead Letter Queues will not be deleted as long as any of their source queues still exist.

## Queue Sharing

Q: How do I share a queue?

A developer associates an access policy statement (specifying the permissions being granted) with the queue to be shared. Amazon SQS provides APIs to create and manage the access policy statements: AddPermission, RemovePermission, SetQueueAttributes and GetQueueAttributes. Refer to the latest API specification for more details.

Q: Who pays for shared queue access?

The queue owner pays for shared queue access.

Q: How do I identify another AWS user?

The Amazon SQS API uses the AWS account number to identify AWS users.

Q: What do I need to provide other users in order to share a queue with them?

You’ll need to provide the full URL from the queue to share. The CreateQueue and ListQueues operations return this URL in their response.

Q: Does Amazon SQS support Anonymous access?

Yes – with the WSDL 2009 and later APIs, a developer can set an access policy that allows anonymous users access to a queue.

Q: When would I use the Permissions API?

The Permissions API provides a simple interface for developers to share access to a queue but it cannot allow for conditional access and more advanced use cases.

Q: When would I use SetQueueAttributes with JSON objects?

The SetQueueAttributes operation supports the full access policy language. Using the policy language, access to a queue can be restricted by IP address and time of day for instance. Refer to the access policy documentation in the latest Developer Guide for more details.

## Service Access and Regions

Q: What are the Amazon SQS service access points in each region?

The US East (Northern Virginia) end-point is: [http://sqs.us-east-1.amazonaws.com](http://sqs.us-east-1.amazonaws.com/)

The US West (Oregon) end-point is: [http://sqs.us-west-2.amazonaws.com](http://sqs.us-west-2.amazonaws.com/)

The US West (Northern California) end-point is: [http://sqs.us-west-1.amazonaws.com](http://sqs.us-west-1.amazonaws.com/)

The EU (Ireland) end-point is: [http://sqs.eu-west-1.amazonaws.com](http://sqs.eu-west-1.amazonaws.com/)

The EU (Frankfurt) end-point is: [http://sqs.eu-central-1.amazonaws.com](http://sqs.eu-central-1.amazonaws.com/)

The Asia Pacific (Singapore) end-point is: [http://sqs.ap-southeast-1.amazonaws.com](http://sqs.ap-southeast-1.amazonaws.com/)

The Asia Pacific (Tokyo) end-point is: [http://sqs.ap-northeast-1.amazonaws.com](http://sqs.ap-northeast-1.amazonaws.com/)

The Asia Pacific (Sydney) end-point is: [http://sqs.ap-southeast-2.amazonaws.com](http://sqs.ap-southeast-2.amazonaws.com/)

The South America (Sao Paulo) end-point is: [http://sqs.sa-east-1.amazonaws.com](http://sqs.sa-east-1.amazonaws.com/)

Q: Can messages be shared between queues in different regions?

No – Amazon SQS in each region is totally independent in message stores and queue names.

Q: Is there a pricing difference between regions?

No – Amazon SQS pricing is the same for all 8 regions.

Q: What is the pricing structure between various regions?

Data transferred between Amazon SQS and Amazon EC2 within a single region is free of charge. Data transferred between Amazon SQS and Amazon EC2 in different regions will be charged at the normal data transfer rate.