Amazon Simple Queue Service

开发人员指南

API Version 2012-11-05

Amazon Web Services

Amazon Simple Queue Service: 开发人员指南

Amazon Web Services

什么是 Amazon Simple Queue Service？

Amazon Simple Queue Service (Amazon SQS) 提供了可靠且可扩展的托管队列，用于存储计算机之间传输的消息。通过使用 Amazon SQS，您可以在执行不同任务的应用程序的分布式组件之间移动数据，既不会丢失消息，也不要求各个组件始终处于可用状态。

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.

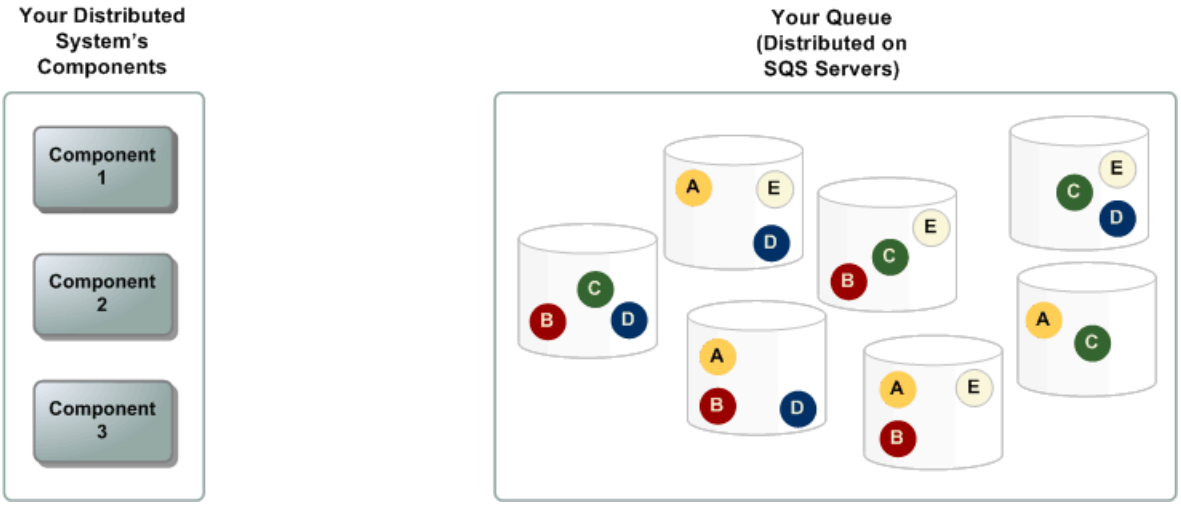
请务必阅读有关分布式队列的信息，这有助于您了解如何设计与 Amazon SQS 正确搭配工作的应用程序。有关更多信息，请参阅 分布式队列的属性 (p. 4)。

架构概述

整个系统中有三个主要的参与者：

- 分布式系统的组件
- 队列
- 队列中的消息

在下图中，您的系统有多个向队列发送消息以及从队列接收消息的组件。该图显示，具有消息（标记为 A-E）的单一队列冗余地保存在多台 Amazon SQS 服务器中。



Amazon SQS功能

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.

Amazon SQS 队列的工作原理

Topics

本部分介绍了 Amazon SQS 队列的基本属性、队列和消息的标识符、确定队列一般大小的方法，以及管理队列中消息的方法。

如果您尚未向队列发送任何消息，或者如果您从队列删除了所有消息，则队列可能为空。

您必须为每个队列指定一个名称（有关更多信息，请参阅“队列 URL (p. 5)”）。您可以获取名称的前几个字符相同的所有队列或队列子集的列表（例如，您可以获取名称以“T3”开头的所有队列的列表）。

无论队列是否为空，您都可以随时删除队列。但请注意，队列会在设置的时间段内保留消息。默认情况下，队列会将消息保留四天。但是，您可以将队列配置为在发送消息后，最多将消息保留 14 天。

如果在连续 30 天内没有对队列执行以下操作之一，则 Amazon SQS 可能会在不进行通知的情况下删除该队列：`SendMessage`、`ReceiveMessage`、`DeleteMessage`、`GetQueueAttributes`、`SetQueueAttributes`、`AddPermission` 和 `RemovePermission`。

Important

如果您重复创建队列，然后让其处于非活跃状态，或者如果您在队列中存储过量数据，则会违背 Amazon SQS 的预期用途。

下表列出了要使用的 API 操作。

要执行以下任务……	请使用以下操作
创建队列	CreateQueue
获取现有队列的 URL	GetQueueUrl
列出队列	ListQueues

要执行以下任务……	请使用以下操作
删除队列	DeleteQueue

分布式队列的属性

以下信息有助于您将您的应用程序设计为与 Amazon SQS 正确地搭配工作。

消息顺序

Amazon SQS 会尽量保持消息顺序，但是由于队列的分布式特性，我们无法保证您将以发送消息的先后顺序接收这些消息。如果您的系统要求保持顺序，则我们建议您在每条消息中放置排序信息，以便在收到消息时对消息重新排序。

至少一次传递

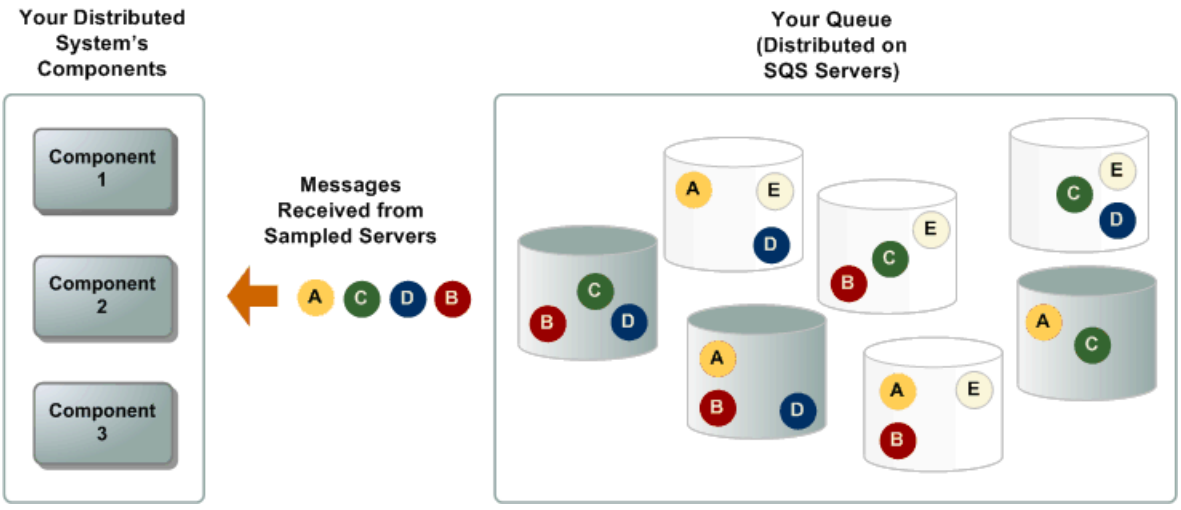
Amazon SQS 会在多台服务器上存储消息的副本，以实现冗余和高可用性。在极少数情况下，当您接收或删除消息时，存储消息副本的服务器之一可能不可用。如果出现这种情况，则该不可用服务器上的消息副本将不会被删除，并且您在接收消息时可能会再次获得该消息副本。因此，您必须将您的应用程序设计为幂等的应用程序（即，如果您的应用程序多次处理同一条消息，则不得受到不利影响）。

消息示例

从队列取回消息的行为取决于您使用的是短（标准）轮询（默认行为）还是长轮询。有关长轮询的更多信息，请参阅“Amazon SQS 长轮询 (p. 10)”。

如果使用短轮询，则您从队列中检索消息时，Amazon SQS 会对服务器的一个子集（基于加权随机分布）进行采样，并且仅从这些服务器返回消息。这意味着，特定接收请求可能不会返回您的所有消息。或者，如果您的队列中有少量消息（少于 1000 条），则意味着，特定请求可能不会返回您的任何消息，而后续请求则会返回您的任何消息。如果您继续从您的队列中检索消息，则 Amazon SQS 会对所有服务器进行采样，您会收到您的所有消息。

下图显示了您的系统组件之一提出接收请求后返回消息的短轮询行为。Amazon SQS 对若干服务器（显示为灰色）进行采样并从这些服务器返回消息（消息 A、C、D 和 B）。系统不会为此特定请求返回消息 E，但会为后续请求返回该消息。



队列和消息标识符

Amazon SQS 使用您需要熟悉的以下三个标识符：

- 队列 URL
- 消息 ID
- 接收句柄

队列 URL

创建新队列时，您必须提供在您的所有队列范围内唯一的队列名称。如果您同时使用最新的 WSDL 和之前的版本创建队列，则您的所有队列仍具有单一命名空间。Amazon SQS 会为您创建的每个队列分配一个名为*队列 URL* 的标识符，该标识符包括队列名称以及 Amazon SQS 确定的其他组件。每当您要对队列执行操作时，都需要提供其队列 URL。

以下是名为“queue2”的队列的队列 URL，该队列由 AWS 账户号为“123456789012”的人员所拥有。

```
http://sqs.us-east-1.amazonaws.com/123456789012/queue2
```

Important

在您的系统中，请始终存储 Amazon SQS 在您创建队列时向您返回的整个队列 URL（例如，`http://sqs.us-east-1.amazonaws.com/123456789012/queue2`）。每当您需要在请求中指定队列 URL 时，请勿从它的各个组件构建队列 URL，原因是 Amazon SQS 可能会更改构成队列 URL 的组件。

此外，您还可以通过列出您的队列来获取队列的队列 URL。即使您的所有队列具有单一命名空间，但是返回的队列列表仍取决于您用于请求的 WSDL。有关更多信息，请参阅“ListQueues”。

消息 ID

每条消息都会收到一个系统分配的*消息 ID*，该 ID 由 Amazon SQS 在 SendMessage 响应中返回给您。此标识符对于识别消息很有用，但是要删除消息，您需要消息的*接收句柄*，而不是消息 ID。消息 ID 的最大长度为 100 个字符。

接收句柄

每当收到来自队列的消息时，您都会收到该消息的*接收句柄*。句柄与接收消息的操作相关联，与消息本身无关。要删除消息或更改消息可见性，您必须提供接收句柄，而不是消息 ID。这意味着，您必须始终先接收消息，然后才能删除它（您不能将消息放入队列中，然后重新调用它）。接收句柄的最大长度为 1 024 个字符。

Important

如果多次接收某条消息，则每次接收该消息时，您都会获得不同的接收句柄。在请求删除该消息时，您必须提供最近收到的接收句柄，否则可能无法删除该消息。

以下是接收句柄的示例。

```
MbZj6wDWli+JvwwJaBV+3dcjk2YW2vA3+STFFljTM8tJJg6HRG6PYSasuWXPJB+Cw
Lj1FjgXUv1uSj1gUPAWV66FU/WeR4mq2OKpEGYWbnLmpRCJVAyeMjeU5ZBdtcQ+QE
auMZc8ZRv37sIW2iJKq3M9MFx1YvV11A2x/KSbkJ0=
```

处理消息所需的资源

为了帮助您估计处理排队消息所需的资源，Amazon SQS 可以为您提供队列中消息的大概数量。您可以查看可见的消息数量，也可以查看不可见的消息数量。有关可见性的更多信息，请参阅“可见性超时 (p. 6)”。

Important

由于 Amazon SQS 采用的是分布式架构，因此，该结果并不是队列中消息数量的准确计数。在大多数情况下，该结果应接近于队列中消息的实际数量，但是您不应指望该计数精确无误。

下表列出了要使用的 API 操作。

要执行以下任务……	请使用以下操作	将 AttributeName 设置为以下值
获取队列中消息的大概数量	GetQueueAttributes	`ApproximateNumberOfMessages`
获取队列中不可见消息的大概数量	GetQueueAttributes	`ApproximateNumberOfMessagesNotVisible`

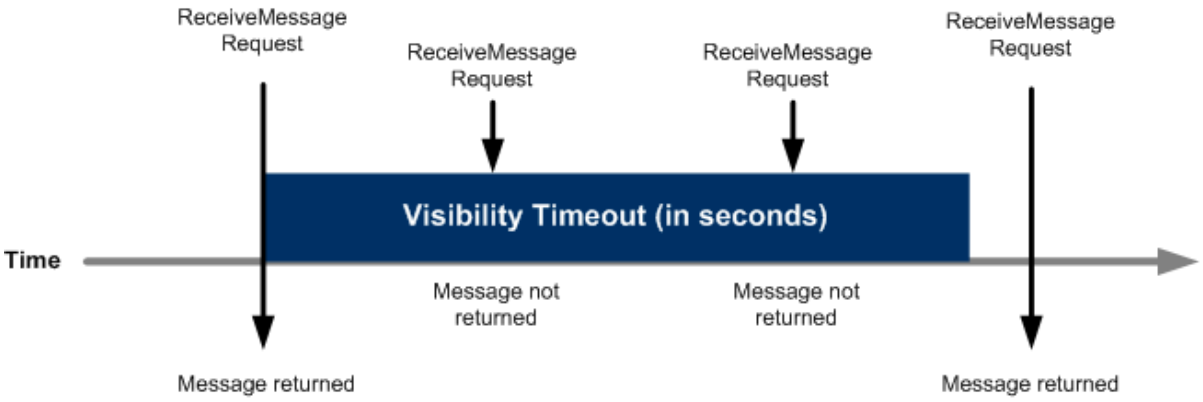
可见性超时

Topics

您系统中的使用组件接收并处理来自队列的消息时，该消息会保留在队列中。为什么 Amazon SQS 不自动删除它？

因为您的系统是分布式系统，所以无法保证该组件会真正收到该消息（在收到该消息前，连接可能会中断，或者该组件可能会发生故障）。因此，Amazon SQS 不会删除该消息，而您的使用组件则必须在收到并处理该消息后将其从队列中删除。

该组件收到该消息后，该消息仍在队列中。但是，您不希望系统中的其他组件再次接收并处理该消息。因此，Amazon SQS 会使用 *可见性超时*（这是 Amazon SQS 阻止其他使用组件接收并处理该消息的一段时间）来阻止这些组件。以下图片和讨论说明了这一概念。



Note

每个队列飞行消息的数量限制为 120,000。消息被队列接收后会处于飞行状态，但尚未从队列中删除。如果您达到 120,000 的限制，将会收到一条来自 Amazon SQS 的“OverLimit”错误消息。为避免达到限制，您应该在处理消息后将其从队列删除。您还可以增加用来处理消息的队列数量。

可见性超时的一般建议

在 Amazon SQS 返回该消息后，可见性超时时钟开始计时。在这段时间里，该组件会处理并删除该消息。但是，如果该组件在删除该消息前发生故障，则会出现什么情况？如果您的系统在可见性超时过期之前没有为该消息调用 DeleteMessage，则该消息会再次变为对您系统中的组件发出的 ReceiveMessage 调用可见，并且会再次被接收。如果某条消息只应被接收一次，则您的系统应在可见性超时期间删除该消息。

每个队列的起始可见性超时的默认设置为 30 秒。您可以为整个队列更改该设置。通常，您会将可见性超时设置为处理消息并从队列中删除消息的平均时间。此外，在接收消息时，您还可以为返回的消息设置特殊的可见性超时，而无需更改整个队列的超时。

如果您有一套生成消息的系统，这些消息需要不同的时间来处理并删除，则我们建议您创建多个队列，并且每个队列具有不同的可见性超时设置。随后，您的系统可以将所有消息发送到单一队列，该队列会根据每条消息的预期处理和删除时间，将该消息转发到具有适当可见性超时的另一个队列。

延长消息的可见性超时

收到来自队列的消息并开始处理该消息时，您可能会发现，该队列的可见性超时不足以完全处理并删除该消息。要为自己提供更多时间来处理该消息，您可以通过使用 `ChangeMessageVisibility` 操作指定新超时值来延长该消息的可见性超时。Amazon SQS 会使用新值重新启动超时期间。

例如，假设该队列的超时为 30 秒，并且您从该队列接收一条消息。在该消息的超时到达 20 秒（即，还剩 10 秒）时，您希望为自己额外提供 60 秒，于是，您立即为该消息调用 `ChangeMessageVisibility`，并将 *`VisibilityTimeout`* 设置为 60 秒。这意味着，您将该消息的可见性超时从 30 秒延长到 80 秒：初始超时设置中的 20 秒加上更改超时后的 60 秒。

延长消息的可见性超时后，新的超时仅会应用于该消息的特定接收。`ChangeMessageVisibility` 不会影响该队列的超时或该消息的后续接收。如果出于某个原因，您没有删除该消息，并且再次接收它，则该消息的可见性超时会是为该队列设置的原始值。

终止消息的可见性超时

收到来自该队列的某条消息时，您可能会发现，您实际上不想处理并删除该消息。Amazon SQS 允许您终止特定消息的可见性超时，这样会立即使该消息对系统中的其他组件可见以得到处理。要执行此操作，您可以调用 ChangeMessageVisibility，并将 `VisibilityTimeout` 设置为 0 秒。

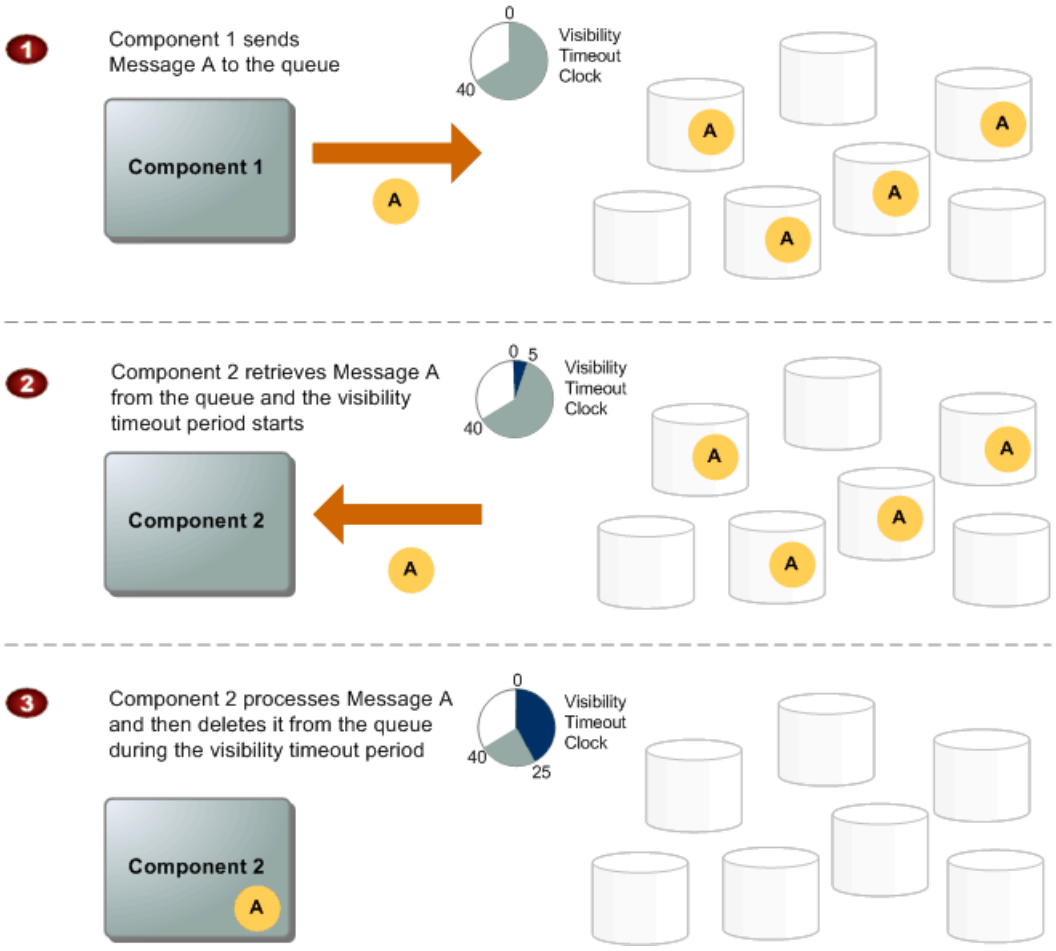
与可见性超时相关的 API 操作

下表列出了用于操纵可见性超时的 API 操作。使用每个操作的 `VisibilityTimeout` 参数来设置或获取值。

要执行以下任务……	请使用以下操作
设置队列的可见性超时	SetQueueAttributes
获取队列的可见性超时	GetQueueAttributes
设置所收到消息的可见性超时，而不影响队列的可见性超时	ReceiveMessage
延长或终止消息的可见性超时	ChangeMessageVisibility
延长或终止多达十条消息的可见性超时	ChangeMessageVisibilityBatch

消息生命周期

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.

Amazon SQS 长轮询

Topics

Amazon SQS 的 API 版本 2012-11-05 提供了对长轮询的支持。

使用 Amazon SQS 进行长轮询的一个好处是：在没有消息可返回以答复发送到 Amazon SQS 队列的 `ReceiveMessage` 请求时，可以减少空响应数量。在发送响应之前，长轮询允许 Amazon SQS 产品等到队列中的消息可用为止。因此，如果连接没有超时，则对 `ReceiveMessage` 请求的响应将至少包含一条可用的消息（如果有），最多可包含 `ReceiveMessage` 调用中请求的最大消息数。

另一个好处是：有助于消除假的空响应（其中，消息在队列中可用，但不包含在响应中）。这种情况在 Amazon SQS 使用短（标准）轮询（默认行为）时会出现，此时，系统只会查询服务器的一个子集（基于加权随机分布），以查看是否有任何消息可包含在响应中。另一方面，如果启用长轮询，则 Amazon SQS 会查询所有服务器。

此外，减少空响应和假空响应的数量还有助于降低使用 Amazon SQS 的费用。

如果 ReceiveMessage 调用的 *`WaitTimeSeconds`* 参数设置为 0，则会出现短轮询。`ReceiveMessage` 这种情况会在以下两种方式之一中出现：`ReceiveMessage` 调用将 *`WaitTimeSeconds`* 设置为 0，或 `ReceiveMessage` 调用不设置 *`WaitTimeSeconds`* 并且队列属性 *`ReceiveMessageWaitTimeSeconds`* 为 0。

Note

为 `ReceiveMessage` 的 *`WaitTimeSeconds`* 参数设置的值（介于 1 到 20 之间）优先于为队列属性 *`ReceiveMessageWaitTimeSeconds`* 设置的任何值。

您可以用来在 Amazon SQS 中启用长轮询的不同的 API 操作调用有三种：`ReceiveMessage`、`CreateQueue` 和 `SetQueueAttributes`。对于 `ReceiveMessage`，您可以配置 `WaitTimeSeconds` 参数；对于 `CreateQueue` 和 `SetQueueAttributes`，您可以配置 `ReceiveMessageWaitTimeSeconds` 属性。

Important

如果您决定对多个队列实施长轮询，则建议您对每个队列使用一个线程，而不是试图使用单一线程来轮询所有队列。如果对每个队列使用一个线程，则您的应用程序可以在各队列中的消息可用

时处理这些消息，这与单一线程用于多个队列的情况相反，在后一种情况下，您的应用程序在等待（最长 20 秒）没有任何可用消息的队列时，可能无法处理其他队列中的可用消息。

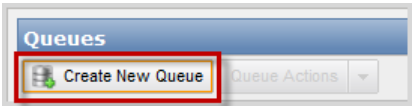
对于大多数使用长轮询的使用案例，您应将超时值设置为最大值（20 秒）。如果最大值（20 秒）不适用于您的应用程序，则您可以选择较短的长轮询超时，最短为 1 秒。如果您不是使用的 AWS SDK 访问 Amazon SQS，或者您已对 AWS SDK 进行了特别配置使其具有较短的超时，则可能需要修改您的 Amazon SQS 客户端，才能允许时间更长的请求或者使用较短的长轮询超时。

使用 AWS Management Console启用长轮询

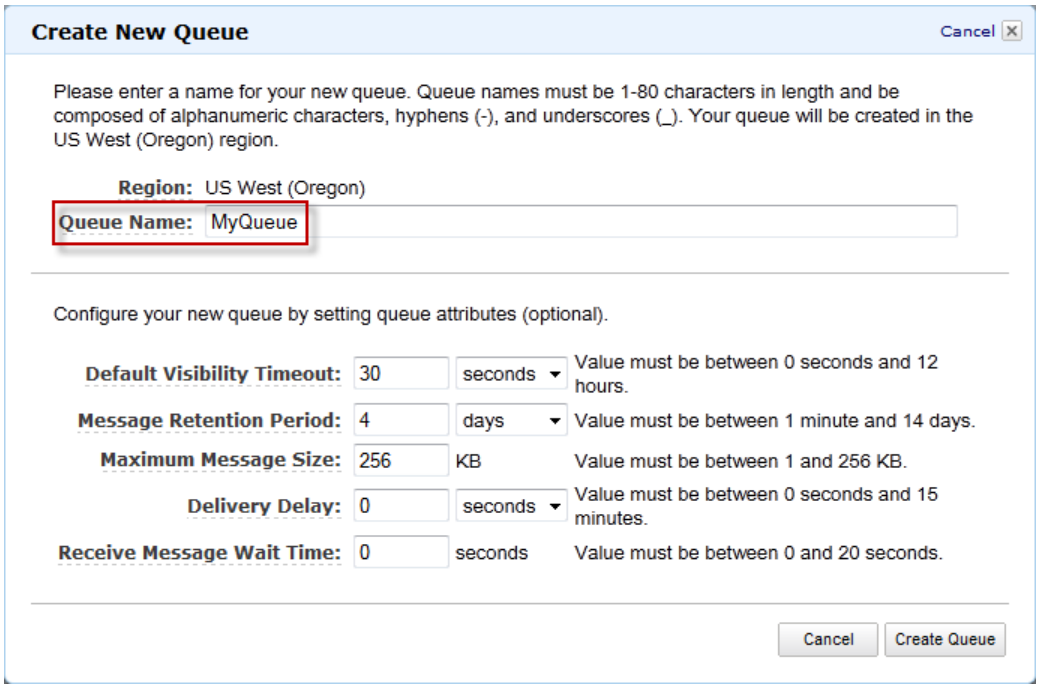
您可以使用 AWS Management Console通过设置大于 0 的 接收消息等待时间 值启用长轮询。

使用 AWS Management Console为新队列启用长轮询

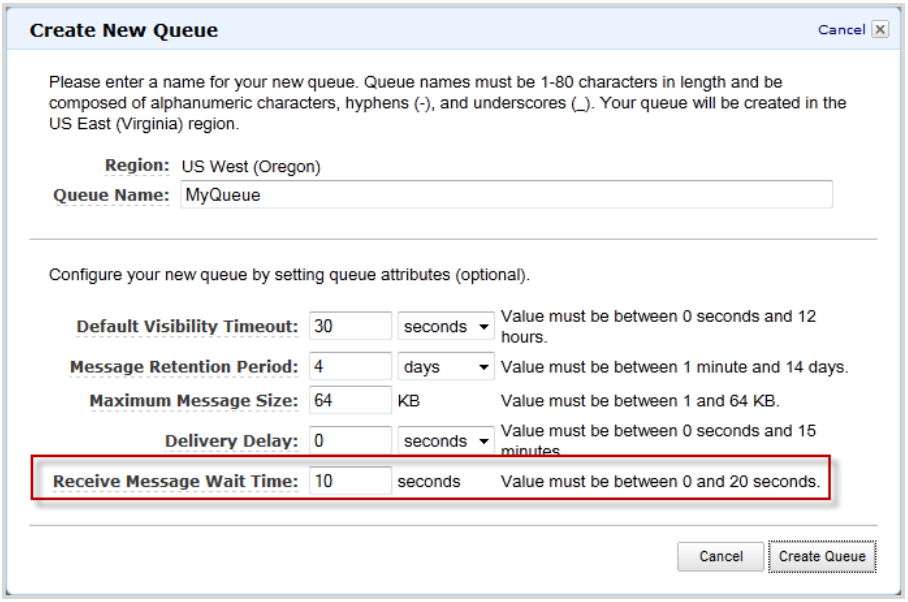
1. 登录 AWS 管理控制台，并通过以下网址打开 Amazon SQS 控制台：https://console.aws.amazon.com/sqs/。
2. 单击 新建队列。



3. 在 新建队列 对话框的 队列名称 字段中，输入队列的名称（例如，MyQueue）。



4. 为 接收消息等待时间 输入介于 1 到 20 秒之间的正整数值。
您可以保留剩余字段的默认值设置，或者输入新值。

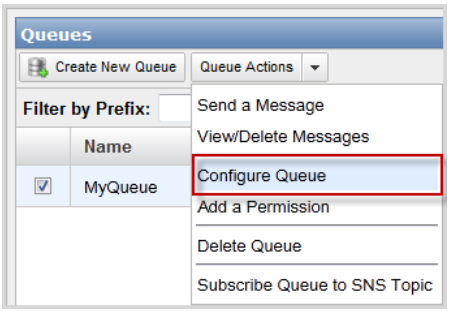


5. 单击 创建队列。

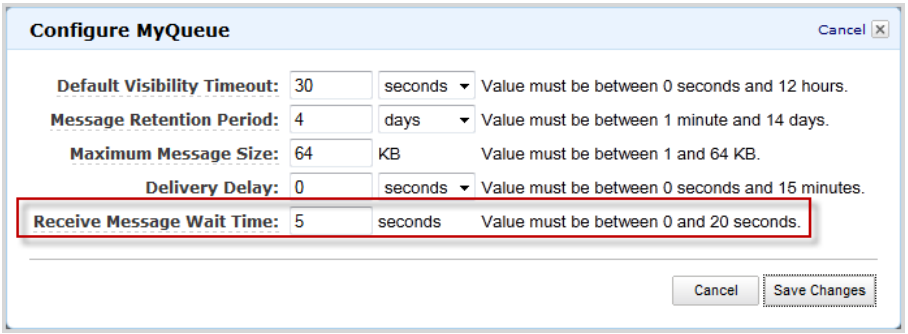
在现有队列突出显示时选择 配置队列 操作，您可以使用 AWS Management Console更改现有队列的 接收消息等待时间 设置。

为现有队列设置新的“接收消息等待时间”值

1. 在现有队列突出显示时选择 配置队列 操作。



2. 将 接收消息等待时间 的值更改为正整数值。



3. 单击 保存更改。

使用查询 API 启用长轮询

以下查询 API 示例通过调用 ReceiveMessage 操作并将 *WaitTimeSeconds* 参数设置为 10 秒来启用长轮询。有关更多信息，请参阅*Amazon Simple Queue Service API 参考*中的“ReceiveMessage”部分。

```
http://sqs.us-east-1.amazonaws.com/123456789012/testQueue/ ?Action=ReceiveMessage
 &WaitTimeSeconds=10 &MaxNumberOfMessages=5 &VisibilityTimeout=15 &Attribute
Name=All; &Version=2012-11-05 &SignatureMethod=HmacSHA256 &Expires=2013-10-
25T22%3A52%3A43PST &AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE &SignatureVersion=2
&Signature=Dqlp3Sd6ljTUA9Uf6SGtEExwUQEXAMPLE
```

以下示例显示了启用长轮询的另一种选择。其中，SetQueueAttributes 操作的 *ReceiveMessageWaitTimeSeconds* 属性设置为 20 秒。有关更多信息，请参阅*Amazon Simple Queue Service API 参考*中的“SetQueueAttributes”部分。

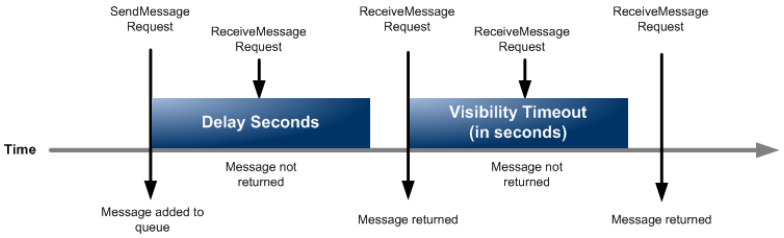
```
http://sqs.us-east-1.amazonaws.com/123456789012/testQueue/ ?Action=SetQueueAt
tributes &Attribute.Name=ReceiveMessageWaitTimeSeconds &Attribute.Value=20
&Version=2012-11-05 &SignatureMethod=HmacSHA256 &Expires=2013-10-
25T22%3A52%3A43PST &AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE &SignatureVersion=2
&Signature=Dqlp3Sd6ljTUA9Uf6SGtEExwUQEXAMPLE
```

Amazon SQS 延迟队列

Topics

延迟队列允许您将队列中新消息的传递操作推迟特定的秒数。如果您创建延迟队列，则发送到该队列的任何消息在延迟期间对使用者都不可见。通过将 *DelaySeconds* 属性设置为介于 0 和 900（15 分钟）之间的任何值，您可以使用 `CreateQueue` 创建延迟队列。此外，通过使用 `SetQueueAttributes` 来设置现有队列的 *DelaySeconds* 属性，您还可以将现有队列转变为延迟队列。

延迟队列类似于可见性超时，因为这两种功能都使得使用者在特定的时间段内无法获得消息。延迟队列和可见性超时之间的区别在于：对于延迟队列，消息在首次添加到队列时是隐藏的；而对于可见性超时，消息只有在从队列取回后才是隐藏的。下图说明了延迟队列和可见性超时之间的关系。



Note

每个队列飞行消息的数量限制为 120,000。消息被队列接收后会处于飞行状态，但尚未从队列中删除。如果您达到 120,000 的限制，将会收到一条来自 Amazon SQS 的“OverLimit”错误消息。为避免达到限制，您应该在处理消息后将其从队列删除。您还可以增加用来处理消息的队列数量。

要设置各条消息（而不是整个队列）的延迟秒数，请使用消息定时器。如果您发送一条带有消息定时器的消息，则 Amazon SQS 会使用消息定时器的延迟秒数值（而不是延迟队列的延迟秒数值）。有关更多信息，请参阅 Amazon SQS 消息定时器 (p. 18)。

Important

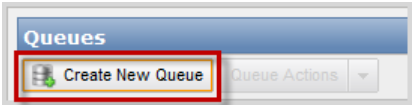
所有 Amazon SQS 队列都已启用延迟功能。您可以将使用 2008-01-01 或 2009-02-01 API 版本创建的任何队列转变为延迟队列。2011-10-01 API 版本和更高的 API 版本中提供了消息定时器。如果您要向各条消息应用特定的延迟值，则必须使用 2011-10-01 API 版本或更高的 API 版本。

使用 AWS Management Console创建延迟队列

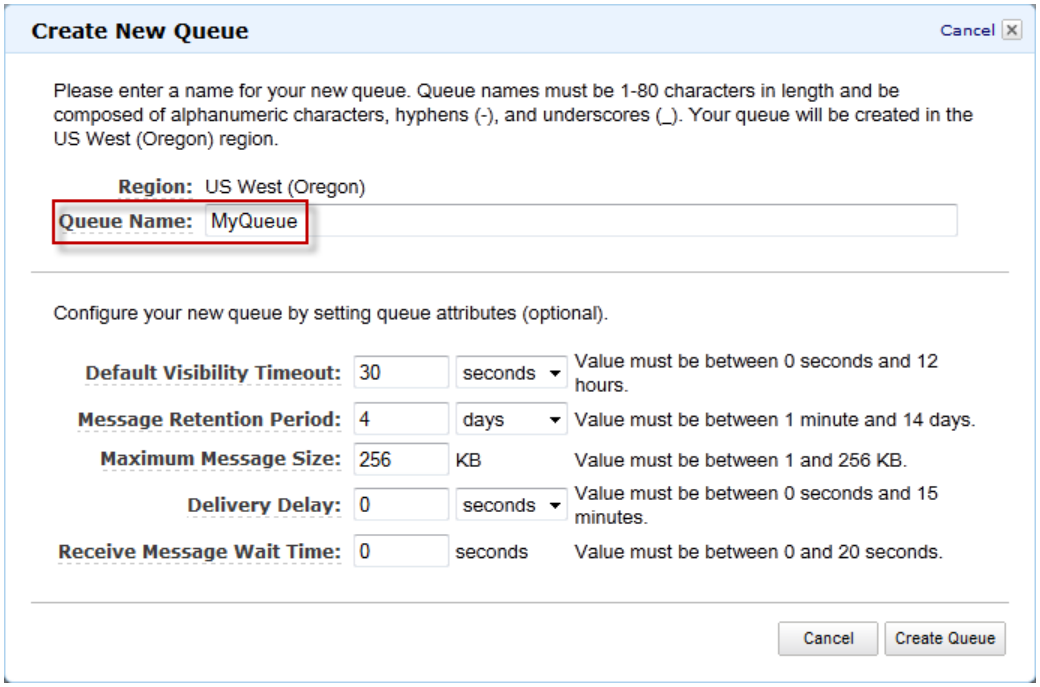
通过设置大于 0 的 交付延迟 值，您可以使用 AWS Management Console创建延迟队列。

使用 AWS Management Console创建延迟队列

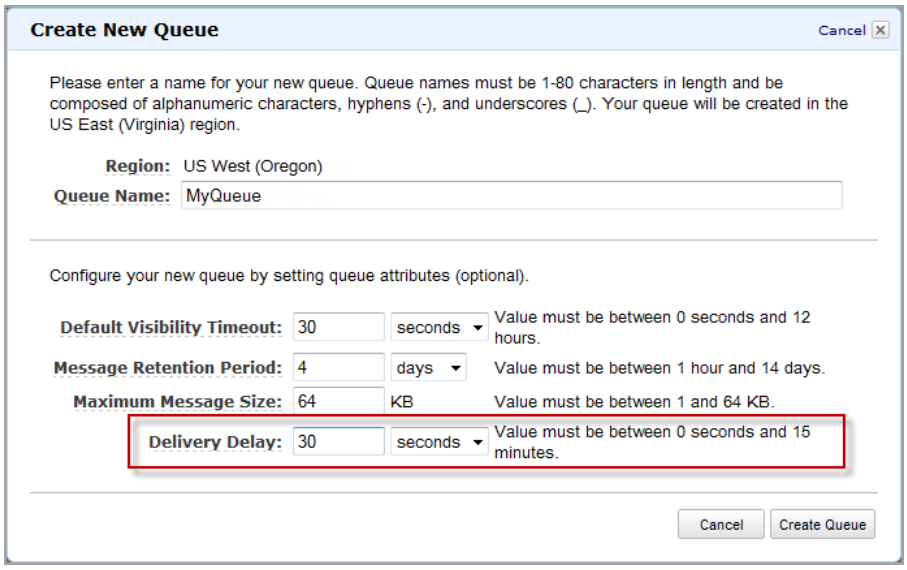
1. 登录 AWS 管理控制台，并通过以下网址打开 Amazon SQS 控制台：https://console.aws.amazon.com/sqs/。
2. 单击 新建队列。



3. 在 新建队列 对话框的 队列名称 字段中，输入队列的名称（例如，MyQueue）。



4. 为“交付延迟”属性输入正整数值（例如，30）。
您可以保留剩余字段的默认值设置，或者输入新值。

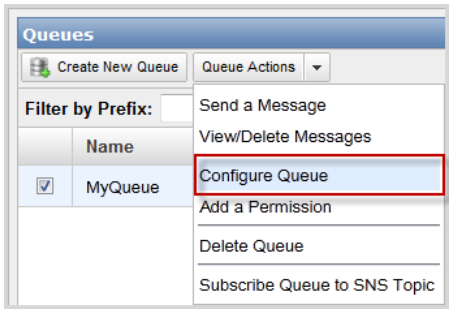


5. 单击 创建队列。

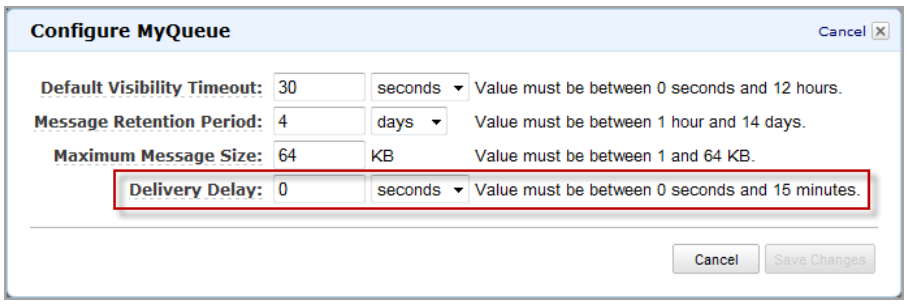
在现有队列突出显示时选择 配置队列 操作，您可以使用 AWS Management Console更改现有队列的 交付延迟 设置。

为现有队列设置新的传递延迟值

1. 在现有队列突出显示时选择 配置队列 操作。



2. 将 交付延迟 的值更改为正整数值。



3. 单击 保存更改。

使用查询 API 创建延迟队列

以下查询 API 示例会调用 `CreateQueue` 操作来创建延迟队列，该延迟队列会在每条消息进入队列后的前 45 秒对使用者隐藏该消息。

```
http://sqs.us-east-1.amazonaws.com/ ?Action=CreateQueue &QueueName=testQueue
&Attribute.1.Name=DelaySeconds &Attribute.1.Value=45 &Version=2011-10-01 &Sig
natureMethod=HmacSHA256 &Expires=2011-10-20T22%3A52%3A43PST &AWSAccessKeyId=AKI
AIOSFODNN7EXAMPLE &SignatureVersion=2 &Signature=Dqlp3Sd6ljTUA9Uf6SGtEEx
wUQEXAMPLE
```

此外，通过将 DelaySeconds 属性从默认值 0 更改为小于或等于 900 的正整数值，您还可以将现有队列更改为延迟队列。以下示例会调用 `SetQueueAttributes` 将名为 `testQueue` 的队列的 *`DelaySeconds`* 属性设置为 45 秒。

```
http://sqs.us-east-1.amazonaws.com/123456789012/testQueue/ ?Action=SetQueueAt
tributes &Attribute.Name=DelaySeconds &Attribute.Value=45 &Version=2011-10-01
&SignatureMethod=HmacSHA256 &Expires=2011-10-20T22%3A52%3A43PST &AWSAccessKey
Id=AKIAIOSFODNN7EXAMPLE &SignatureVersion=2 &Signature=Dqlp3Sd6ljTUA9Uf6SGtEEx
wUQEXAMPLE
```

Amazon SQS 消息定时器

Topics

Amazon SQS 消息定时器允许您为要添加到队列的消息指定初始不可见时段。例如，如果您发送一条消息并将 *DelaySeconds* 参数设置为 45，则使用者在该消息进入队列后的前 45 秒将看不到该消息。`DelaySeconds` 的默认值为 0。

Note

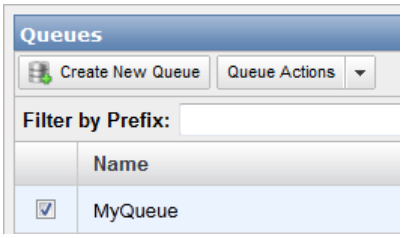
每个队列飞行消息的数量限制为 120,000。消息被队列接收后会处于飞行状态，但尚未从队列中删除。如果您达到 120,000 的限制，将会收到一条来自 Amazon SQS 的“OverLimit”错误消息。为避免达到限制，您应该在处理消息后将其从队列删除。您还可以增加用来处理消息的队列数量。

要设置适用于队列中所有消息的延迟时段，请使用延迟队列。有关更多信息，请参阅 Amazon SQS 延迟队列 (p. 14)。各条消息的消息定时器设置会覆盖适用于整个延迟队列的任何 `DelaySeconds` 值。

使用控制台创建消息定时器

使用 AWS Management Console发送带有消息定时器的消息

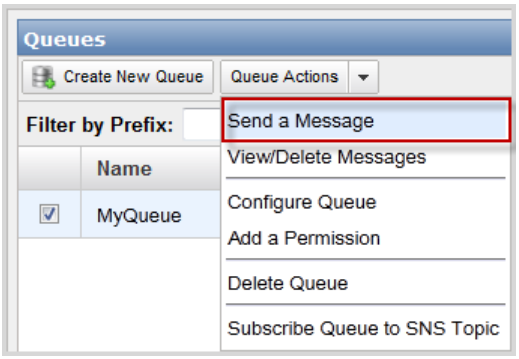
1. 登录 AWS 管理控制台，并通过以下网址打开 Amazon SQS 控制台：https://console.aws.amazon.com/sqs/。
2. 选择队列。



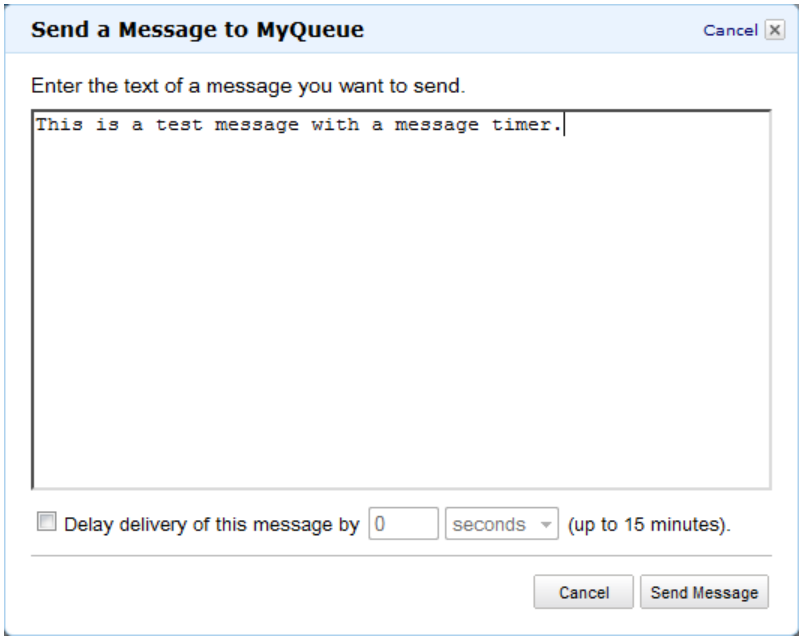
3. 从 队列操作 下拉列表中选择 发送消息。

   Note

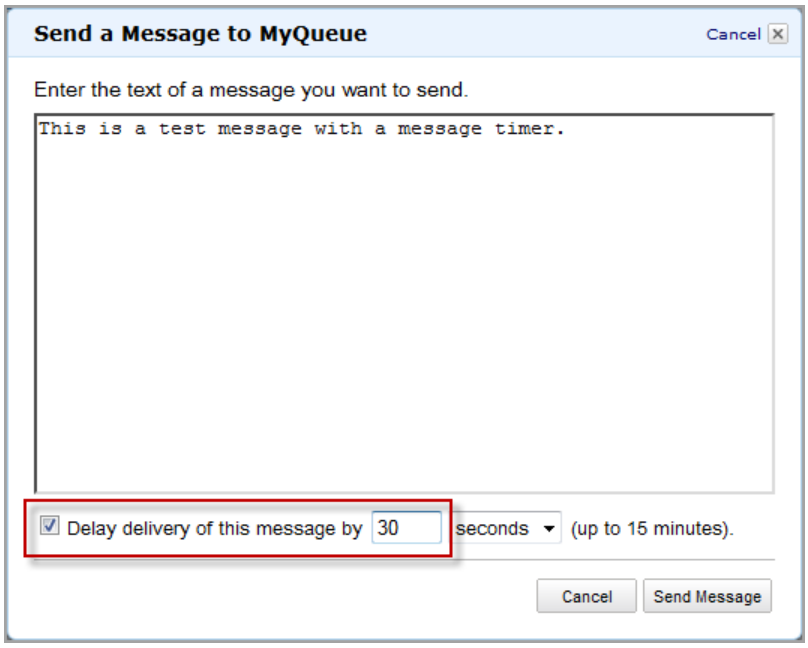
   仅在已选中队列的情况下，队列操作 下拉列表才可用。



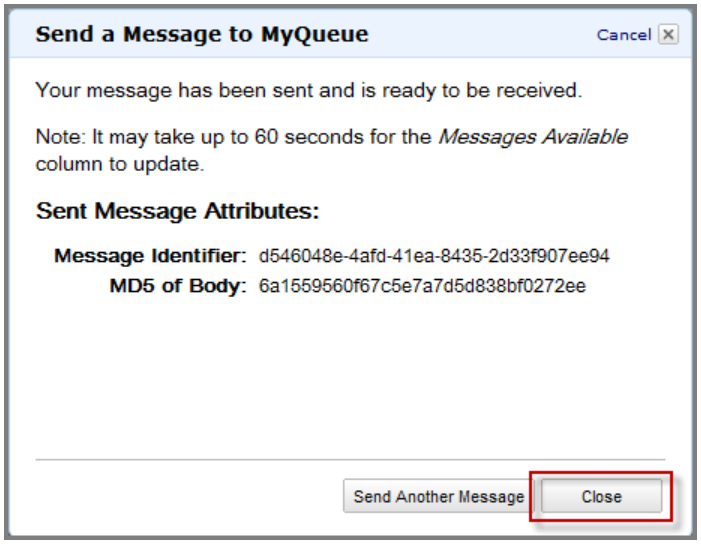
4. 在 发送消息至 MyQueue 对话框中，输入消息（例如，`This is a test message with a message timer.`）。



5. 在 该消息的延迟交付原因 文本框中，输入延迟值（例如，`30`）。



6. 单击 发送消息。
7. 在 发送消息至 MyQueue 确认框中，单击 关闭。



使用查询 API 创建消息定时器

以下查询 API 示例会为使用 SendMessage 发送的单一消息应用 45 秒的初始可见性延迟。

```
http://sqs.us-east-1.amazonaws.com/123456789012/testQueue/ ?Action=SendMessage
 &MessageBody=This+is+a+test+message &Attribute.Name=DelaySeconds &Attrib
ute.Value=45 &Version=2011-10-01 &SignatureMethod=HmacSHA256 &Expires=2011-10-
```

```
18T22%3A52%3A43PST &AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE &SignatureVersion=2
&Signature=Dqlp3Sd6ljTUA9Uf6SGtEExwUQEXAMPLE
```

此外，您还可以使用查询 API 的 SendMessageBatch 操作来发送最多十条带有消息定时器的消息。您可以为每条消息分配不同的 *DelaySeconds* 值，或者完全不分配任何值。如果您不为 DelaySeconds 设置值，则将消息添加到延迟队列时，消息仍然可能会延迟。有关延迟队列的更多信息，请参阅“Amazon SQS 延迟队列 (p. 14)”。以下示例使用 SendMessageBatch 发送三条消息：一条不带消息定时器的消息，以及两条具有不同的 *DelaySeconds* 值的消息。

```
http://sqs.us-east-1.amazonaws.com/123456789012/testQueue/ ?Action=SendMessage
Batch &SendMessageBatchRequestEntry.1.Id=test_msg_no_message_timer &SendMessage
BatchRequestEntry.1.MessageBody=test%20message%20body%201 &SendMessageBat
chRequestEntry.2.Id=test_msg_delay_45_seconds &SendMessageBatchRequestEntry.2.Mes
sageBody=test%20message%20body%202 &SendMessageBat
chRequestEntry.2.DelaySeconds=45 &SendMessageBat
chRequestEntry.3.Id=test_msg_delay_2_minutes &SendMessageBatchRequestEntry.3.Mes
sageBody=test%20message%20body%203 &SendMessageBat
chRequestEntry.3.DelaySeconds=120 &Version=2011-10-01 &SignatureMethod=HmacSHA256
 &Expires=2011-10-18T22%3A52%3A43PST &AWSAccessKeyId=AKIAI44QH8DHBEXAMPLE
&SignatureVersion=2 &Signature=Dqlp3Sd6ljTUA9Uf6SGtEExwUQEXAMPLE
```

Amazon SQS 批处理 API 操作

Topics

在 Amazon SQS 的 2009-02-01 API 版本中，只有一个操作—ReceiveMessage—支持批处理，即，使用单一调用处理多条消息。在 2011-10-01 API 版本中，Amazon SQS 添加了用于发送消息、删除消息以及更改消息可见性超时值的批处理功能。要同时发送最多十条消息，请使用 SendMessageBatch 操作。要使用一次 API 调用删除最多十条消息，请使用 DeleteMessageBatch 操作。要更改最多十条消息的可见性超时值，请使用 ChangeMessageVisibilityBatch 操作。

要使用新的批处理操作，您必须使用查询 API 或支持新批处理操作的软件开发工具包。请查看您的特定 SDK 的文档，以了解它是否支持新的 Amazon SQS 批处理操作。Amazon SQS 控制台当前不支持批处理 API 操作。

有关这三个批处理 API 操作的详细信息和示例，请转到*Amazon Simple Queue Service API 参考*：

- ChangeMessageVisibilityBatch
- DeleteMessageBatch
- SendMessageBatch

SendMessageBatch 的最大消息大小

您可以使用 SendMessageBatch 发送一条最长为 262,144 字节 (256 KB) 的消息。但是，您在对 SendMessageBatch 的单次调用中发送的所有消息的总大小不能超过 262,144 字节 (256 KB)。

创建 API 请求

Topics

本部分介绍了如何向 Amazon SQS 发出请求。这些主题会帮助您熟悉界面之间的基本区别、请求的组件、对请求进行身份验证的方法，以及响应的内容。

此外，我们还提供了 SDK，这些 SDK 可让您通过您的首选编程语言访问 Amazon SQS。这些开发工具包包含自动处理诸如以下任务的功能：

- 使用密码对服务请求签名
- 重试请求
- 处理错误响应

有关可用的 SDK 的列表，请转到“适用于 Amazon Web Services 的工具”

终端节点

有关此产品地区和终端节点的信息，请访问《Amazon Web Services General Reference》中的“地区和终端节点”部分。

例如，要在欧洲创建队列，您可能会生成类似以下请求的查询请求：

```
   http://sqs.eu-west-1.amazonaws.com/ ?Action=CreateQueue &DefaultVisibility
Timeout=40 &QueueName=testQueue &Version=2009-02-01 &SignatureMethod=HmacSHA256
 &Expires=2009-04-18T22%3A52%3A43PST &AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&SignatureVersion=2 &Signature=Dqlp3Sd6ljTUA9Uf6SGtEExwUQEXAMPLE
```

每个 Amazon SQS 终端节点完全独立。例如，如果您有两个名为“MyQueue”的队列，一个在 sqs.us-east-1.amazonaws.com 中，一个在 sqs.eu-west-1.amazonaws.com 中，则它们是完全独立的，不共享任何数据。

提出查询请求

Topics

Amazon SQS 支持用于调用服务操作的查询请求。查询请求是使用 GET 或 POST 方法的简单 HTTP 或 HTTPS 请求。查询请求必须包含 *Action* 参数，以指示要执行的操作。响应是符合某种模式的 XML 文档。

GET 请求的结构

本指南将 Amazon SQS GET 请求显示为可直接在浏览器中使用的 URL。URL 由以下要素构成：

- 终端节点—请求对其执行操作的资源（在使用 Amazon SQS 时，终端节点是队列）
- 操作—您希望对终端节点执行的操作；例如：发送消息
- 参数—任何请求参数

以下是向 Amazon SQS 队列发送消息的示例 GET 请求。

```
http://sqs.us-east-1.amazonaws.com/123456789012/queue1?Action=SendMessage&Mes
sageBody=Your%20Message%20Text&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Version=2012-
11-05&Expires=2008-02-10T12:00:00Z&Signature=lBP67vCvGlDMBQ1do
fZxg8E8SUEXAMPLE&SignatureVersion=2&SignatureMethod=HmacSHA256
```

Important

由于 GET 请求是 URL，因此，您必须对参数值进行 URL 编码。例如，在前面的示例请求中，*MessageBody* 参数的值实际为 `Your Message Text`。但是，URL 中不允许使用空格，因此 URL 中的空格均编码为“`%20`”。示例的其余部分没有进行 URL 编码，更易于您阅读。

为了使 GET 示例更易于阅读，本指南采用了以下解析格式来显示这些示例。

```
http://sqs.us-east-1.amazonaws.com/123456789012/queue1 ?Action=SendMessage
&MessageBody=Your%20Message%20Text &AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE &Ver
sion=2012-11-05 &Expires=2011-10-15T12:00:00Z &Signature=lBP67vCvGlDM
BQ1dofZxg8E8SUEXAMPLE &SignatureVersion=2 &SignatureMethod=HmacSHA256
```

Note

在本指南中提供的示例查询请求中，我们使用了假 AWS 访问密钥 ID 和假签名，各项均附加有 `EXAMPLE`。我们这样做是为了表明，您不应根据示例中显示的请求参数来指望示例中的签名准确无误。针对创建查询请求签名的说明中有一个例外。该示例显示了基于我们指定的特定 AWS 访问密钥 ID 以及示例中请求参数的真实签名（有关更多信息，请参阅“查询请求身份验证 (p. 34)”）。

在 Amazon SQS 中，除 *MessageBody* 以外的所有参数始终具有不带空格的值。您为 SendMessage 请求中的 *MessageBody* 提供的值可以包含空格。在本指南中，带有包含空格的 *MessageBody* 的任何示例 `SendMessage` 查询请求在显示时，其中的空格在 URL 中都进行了编码（编码为 `%20`）。为了清楚起见，URL 的其余部分没有使用 URL 编码格式显示。

第一行代表请求的*终端节点*。这是该请求所影响的资源。前面的示例对队列执行操作，因此，请求的终端节点是队列的标识符，称为*队列 URL*。有关队列 URL 的更多详细信息，请参阅“队列 URL (p. 5)”。

终端节点后是一个问号（?），将终端节点与参数隔开。参数之间用“&”号隔开。

`Action` 参数表示要执行的操作（有关操作的列表，请参阅《Amazon SQS API 参考》中的“API 操作”部分）。有关所有查询请求常用的其他参数的列表，请参阅《Amazon SQS API 参考》中的“所有操作常用的请求参数”部分。

POST 请求的结构

Amazon SQS 也接受 POST 请求。使用 POST 请求，您可以在 HTTP 请求正文中以表单发送查询参数，如以下步骤所述。

创建 POST 请求

1. 将查询参数名称和值汇编到表单中。

   这意味着，您可以像对待 GET 请求那样，将参数和值放在一起（使用 & 号分隔每个名称-值对）。以下示例显示了带有换行符的 `SendMessage` 请求；我们在本指南中使用这些换行符是为了使信息更易于阅读。

   ```
   Action=SendMessage &MessageBody=Your Message Text &AWSAccessKeyId=AKIAIOS
   FODNN7EXAMPLE &Version=2012-11-05 &Expires=2011-10-15T12:00:00Z &Signature
   Version=2 &SignatureMethod=HmacSHA256
   ```

2. 根据 HTML 规范的“*表单提交*”一节，对表单进行表单 URL 编码（有关更多信息，请转到 http://www.w3.org/MarkUp/html-spec/html-spec_toc.html#SEC8.2.1）。

   ```
   Action=SendMessage &MessageBody=Your+Message+Text &AWSAccessKeyId=AKIAIOS
   FODNN7EXAMPLE &Version=2012-11-05 &Expires=2011-10-15T12%3A00%3A00Z &Signa
   tureVersion=2 &SignatureMethod=HmacSHA256
   ```

3. 向表单添加请求签名（有关更多信息，请参阅“查询请求身份验证 (p. 34)”）。

   ```
   Action=SendMessage &MessageBody=Your+Message+Text &AWSAccessKeyId=AKIAIOS
   FODNN7EXAMPLE &Version=2012-11-05 &Expires=2011-10-15T12%3A00%3A00Z &Signa
   tureVersion=2 &SignatureMethod=HmacSHA256 &Signature=lBP67vCvGlDM
   BQ1dofZxg8E8SUEXAMPLE
   ```

4. 提供结果表单作为 POST 请求的正文。
5. 包含 `Content-Type` HTTP 标头，并将值设置为 `application/x-www-form-urlencoded`。

以下示例显示了最终的 POST 请求。

```
POST /queue1 HTTP/1.1 Host: sqs.us-east-1.amazonaws.com Content-Type: applica
tion/x-www-form-urlencoded Action=SendMessage &MessageBody=Your+Message+Text
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE &Version=2012-11-05 &Expires=2011-10-
15T12%3A00%3A00Z &SignatureVersion=2 &SignatureMethod=HmacSHA256 &Signa
ture=lBP67vCvGlDMBQ1dofZxg8E8SUEXAMPLE
```

除了 `Content-Type` 以外，Amazon SQS 不要求在请求中使用其他 HTTP 标头。您提供的身份验证签名是您在发送 GET 请求时所提供的签名（有关签名的信息，请参阅“查询请求身份验证 (p. 34)”）。

Note

根据您 HTTP 客户端所使用的 HTTP 版本的需要，该客户端通常会向 HTTP 请求添加其他项目。我们没有在本指南的示例中包含这些额外的项目。

相关主题

- 查询请求身份验证 (p. 34)
- 响应 (p. 35)

提出 SOAP 请求

Important

自 2011 年 8 月 8 日起，Amazon SQS 不再支持 SOAP 请求。

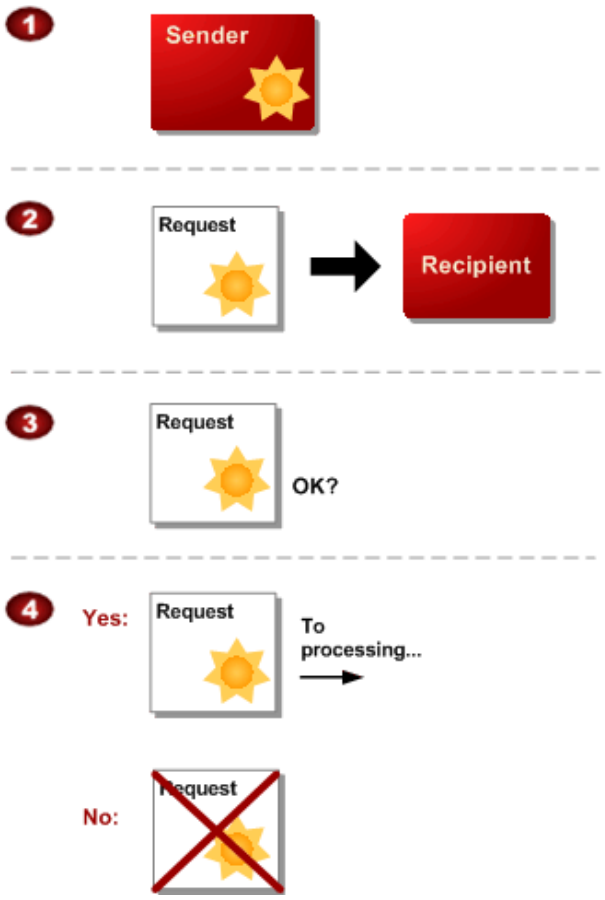
请求身份验证

Topics

本部分中的主题介绍了 Amazon SQS 如何对您的请求进行身份验证。在本节中，您可以了解身份验证的基础知识、您的 AWS 账户和标识符如何用于支持身份验证，以及如何创建 HMAC-SHA1 签名。此外，本节还介绍了查询请求的请求身份验证要求。

什么是身份验证？

身份验证是一个用于识别并验证发送请求的用户的过程。下图显示了简化版的身份验证过程。



身份验证的一般过程

1	发件人获取必要的证书。

2	发件人向收件人发送带有证书的请求。
3	收件人使用证书来验证发件人是否确实发送了该请求。
4	如果是，则收件人会处理该请求。否则，收件人会拒绝该请求并相应地作出响应。

在身份验证过程中，AWS 会同时验证发件人的标识以及发件人是否已注册使用 AWS 提供的服务。如果任一测试失败，则该请求不会得到进一步处理。

有关身份验证的进一步讨论，请转到 techencylopedia.com 的身份验证条目。有关与身份验证相关的常用行业术语的定义，请转到“RSA 实验室术语表”。

后续几部分介绍了 Amazon SQS 如何实施身份验证来保护您的数据。

您的 AWS 账户

您必须首先在 http://aws.amazon.com 上创建 AWS 账户，然后才能访问 AWS 提供的任何 Web 服务。AWS 账户只是可以使用 AWS 产品的 Amazon.com 账户；您可以在创建 AWS 账户时使用现有的 Amazon.com 账户登录名和密码。

或者，您可以通过使用新的登录名和密码来创建一个启用了 AWS 的新 Amazon.com 账户。您提供作为账户登录名的电子邮件地址必须是有效的。系统会要求您提供信用卡或其他付款方式，以支付您使用的任何 AWS 产品的费用。

您可以从您的 AWS 账户查看 AWS 账户活动和使用率报告，以及管理 AWS 账户访问标识符。

相关主题

- 您的 AWS 标识符 (p. 29)

您的 AWS 标识符

在创建 AWS 账户时，AWS 会向您分配一对相关的标识符：

- 访问密钥 ID（20 位字符的字母数字序列）
  例如：AKIAIOSFODNN7EXAMPLE
- 私有访问密钥（40 位字符序列）
  例如：# wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY

这些是您的 *AWS 访问密钥标识符*。

Caution

您的私有访问密钥是秘密，只有您和 AWS 才应该知道。请注意对该密钥保密，以保护您的账户。请勿将其包括在您向 AWS 提出的请求中，也不要通过电子邮件发送给任何人。请勿对组织外部共享密钥，即使有来自 AWS 或 Amazon.com 的询问。合法代表亚马逊的任何人永远都不会要求您提供私有访问密钥。

访问密钥 ID 与您的 AWS 账户关联。您需要将其包括在 AWS 服务请求中，表明您是请求的发送者。

访问密钥 ID 不是秘密，任何人均可在向 AWS 提出的请求中使用您的访问密钥 ID。此外，为了证明您确实是请求的发件人，您还必须包含数字签名。对于所有请求，您可以使用您的私有访问密钥来计算签名。

AWS 会使用请求中的访问密钥 ID 来查找您的私有访问密钥，然后使用该密钥计算数字签名。如果 AWS 计算的签名与您发送的签名相匹配，则请求会被认为是真实的。否则，请求的身份验证将失败并且请求不会被处理。

相关主题

- HMAC-SHA 签名 (p. 30)
- 查询请求身份验证 (p. 34)

查看您的 AWS 标识符

您的访问密钥 ID 和私有访问密钥在您创建 AWS 账户时向您显示，而不会通过电子邮件发送给您。如果需要再次查看，您可以从 AWS 账户看到它们。

查看您的 AWS 访问证书的步骤

1. 登录到 AWS 安全证书网站。
2. 向下滚动到 访问证书 部分，并选择 访问密钥 选项卡。
3. 在 您的访问密钥 列表中查找有效的访问密钥。
4. 要显示私有访问密匙，请单击 私有访问密钥 一栏中的 显示。

相关主题

- 您的 AWS 账户 (p. 29)
- 您的 AWS 标识符 (p. 29)

HMAC-SHA 签名

Topics

- 所需的身份验证信息 (p. 30)
- 基本身份验证过程 (p. 32)
- 关于待签字符串 (p. 33)
- 关于时间戳 (p. 33)
- Base64 编码的 Java 示例代码 (p. 34)
- 计算 HMAC-SHA1 签名的 Java 示例代码 (p. 34)

本部分中的主题介绍了 Amazon SQS 如何使用 HMAC-SHA 签名对查询请求进行身份验证。

所需的身份验证信息

使用查询 API 访问 Amazon SQS 时，要对请求进行身份验证，必须提供以下项目：

- AWS 访问密钥 ID—您的 AWS 账户由您的访问密钥 ID 标识，AWS 使用该 ID 来查找您的私有访问密匙。
- 签名—每个请求都必须包含一个有效的 HMAC-SHA 请求签名，否则请求会被拒绝。
  您通过使用您的私有访问密钥（该密钥是只有您和 AWS 才知道的共享秘密）来计算请求签名。
- 日期—每个请求都必须包含请求的时间戳。您可以为请求提供过期日期和时间来代替时间戳或者作为时间戳的补充。

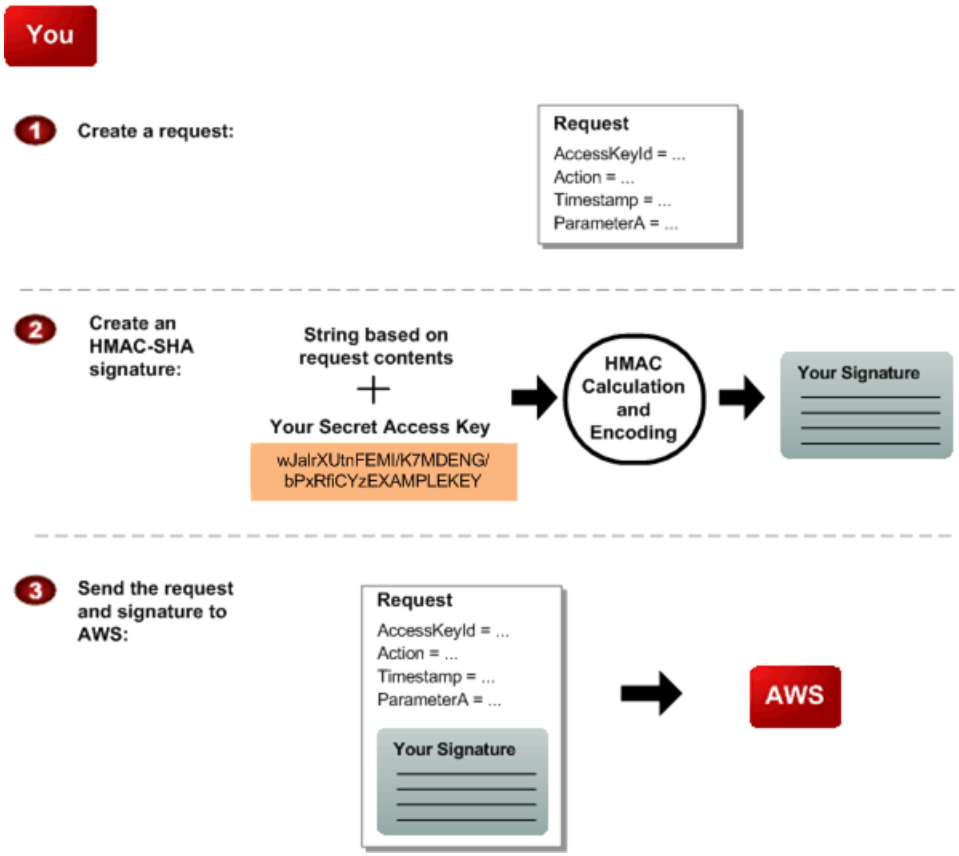
相关主题

- 您的 AWS 标识符 (p. 29)

基本身份验证过程

以下是使用 HMAC-SHA 请求签名对发送到 AWS 的请求进行身份验证所需执行的一系列任务。假设您已经创建了 AWS 账户，并且创建了访问密钥 ID 和私有访问密匙。有关 AWS 账户、访问密钥 ID 和私有访问密匙的更多信息，请参阅“您的 AWS 账户 (p. 29)”和“您的 AWS 标识符 (p. 29)”。

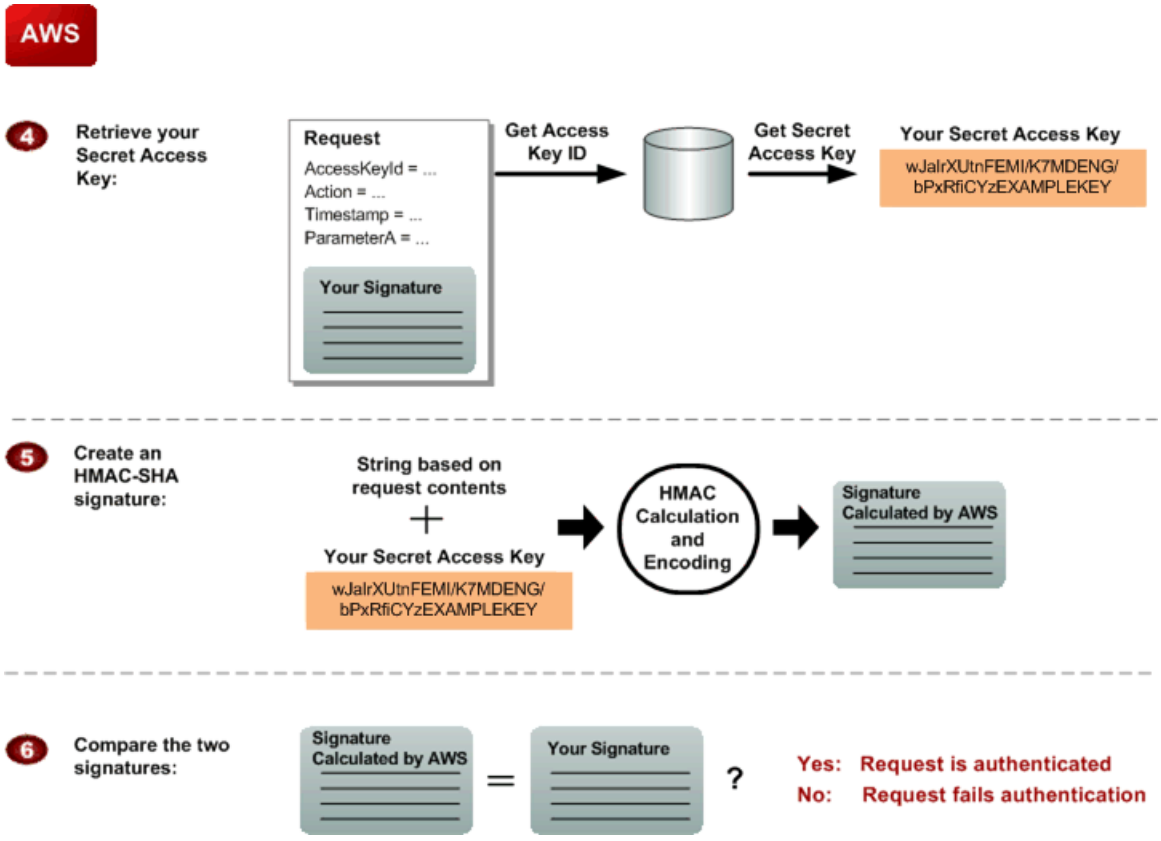
您执行前三个任务。



身份验证过程：您执行的任务

1	您构造发送到 AWS 的请求。
2	您使用您的私有访问密匙来计算密钥式哈希消息身份验证代码 (HMAC-SHA) 签名（有关 HMAC 的信息，请转到 http://www.faqs.org/rfcs/rfc2104.html ）
3	您在请求中包含签名和您的访问密钥 ID，然后将请求发送到 AWS。

AWS 执行后三个任务。



身份验证过程：AWS 执行的任务

4	AWS 使用访问密钥 ID 来查找私有访问密钥。
5	通过使用与计算请求中发送的签名相同的算法，AWS 会从请求数据和私有访问密钥中生成一个签名。
6	如果由 AWS 生成的签名与您在请求中发送的签名相匹配，将认为请求是真实的。如果比较签名这一操作失败，那么请求将被丢弃，同时 AWS 将返回一份错误响应。

关于待签字符串

您发送的每个 AWS 请求都必须包含一个使用您的私有访问密钥计算的 HMAC-SHA 请求签名。详细信息可在“查询请求身份验证 (p. 34)”中找到。

关于时间戳

您在请求中使用的时间戳（或过期时间）必须是 `dateTime` 数据元，并且带有完整的日期和时分秒（有关更多信息，请转到 http://www.w3.org/TR/xmlschema-2/#dateTime）。例如：2007-01-31T23:59:59Z. 我们建议您以协调世界时（格林威治标准时间）时区提供时间戳（虽然这并不是必须的）。

如果您指定了时间戳（而不是过期时间），则请求会在时间戳过后 15 分钟自动过期（换句话说，如果请求的时间戳比 AWS 服务器上的当前时间早 15 分钟以上，则 AWS 不会处理请求）。确保您的服务器时间设置正确。

Important

如果您使用的是 .NET，则不能发送过于特定的时间戳，因为 AWS 和 .NET 对如何确定额外时间精度有不同的解释。要避免过于具体的时间戳，请手动构造精度不超过毫秒的 `dateTime` 数据元。

Base64 编码的 Java 示例代码

请求签名必须采用 Base64 编码。以下 Java 示例代码显示了如何执行 Base64 编码。

```
package amazon.webservices.common; /** * This class defines common routines for
 encoding data in AWS requests. */ public class Encoding { /** * Performs base64-
encoding of input bytes. * * @param rawData * Array of bytes to be encoded. *
@return * The base64 encoded string representation of rawData. */ public static
 String EncodeBase64(byte[] rawData) { return Base64.encodeBytes(rawData); } }
```

计算 HMAC-SHA1 签名的 Java 示例代码

以下 Java 代码示例显示了如何计算 HMAC 请求签名。

```
package amazon.webservices.common; import java.security.SignatureException;
import javax.crypto.Mac; import javax.crypto.spec.SecretKeySpec; /** * This
class defines common routines for generating * authentication signatures for
AWS requests. */ public class Signature { private static final String
HMAC_SHA1_ALGORITHM = "HmacSHA1"; /** * Computes RFC 2104-compliant HMAC signa
ture. * * @param data * The data to be signed. * @param key * The signing key.
 * @return * The Base64-encoded RFC 2104-compliant HMAC signature. * @throws *
 java.security.SignatureException when signature generation fails */ public
static String calculateRFC2104HMAC(String data, String key) throws java.secur
ity.SignatureException { String result; try { // get an hmac_sha1 key from the
 raw key bytes SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(),
HMAC_SHA1_ALGORITHM); // get an hmac_sha1 Mac instance and initialize with the
 signing key Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM); mac.init(sign
ingKey); // compute the hmac on input data bytes byte[] rawHmac = mac.doFi
nal(data.getBytes()); // base64-encode the hmac result = Encoding.Encode
Base64(rawHmac); } catch (Exception e) { throw new SignatureException("Failed
to generate HMAC : " + e.getMessage()); } return result; } }
```

查询请求身份验证

以编程方式调用 Amazon SQS API 提供的功能时，所有发送到 Amazon SQS 的调用都必须经过签名。如果您使用 AWS SDK，则 SDK 会为您处理签名过程，因此您不必手动完成这些任务。另一方面，如果您通过 HTTP/HTTPS 提交查询请求，则必须在每个查询请求中包含签名。

Amazon SQS 支持签名版本 2 和签名版本 4。签名版本 4 提供改进的安全性和性能。如果您正在创建使用 Amazon SQS 的新应用程序，则应使用签名版本 4。

有关如何使用签名版本 4 创建签名的信息，请参阅《AWS General Reference》中的“签名版本 4 签名过程”部分。

有关如何使用签名版本 2 创建签名的信息，请参阅《AWS General Reference》中的“签名版本 2 签名过程”部分。

响应

Topics

在响应操作请求时，Amazon SQS 会返回包含请求结果的 XML 数据结构。此数据符合 Amazon SQS 架构。有关更多信息，请参阅《Amazon SQS API 参考》中的“WSDL 位置和 API 版本”部分。

成功响应的结构

如果请求成功了，则主要响应元素会以该操作命名，但会附加上“Response”。例如，`CreateQueueResponse` 是针对成功的 `CreateQueue` 请求返回的响应元素。此元素包含以下子元素：

- `ResponseMetadata`，其中包含 `RequestId` 子元素
- 包含特定于操作的结果的可选元素；例如，`CreateQueueResponse` 元素包含名为 `CreateQueueResult` 的元素

XML 架构描述了针对每个 SQS 操作的 Amazon SQS 响应消息。

以下是成功响应的示例。

```
<CreateQueueResponse xmlns=http://sqs.us-east-1.amazonaws.com/doc/2012-11-05/
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:type=CreateQueueResponse>
 <CreateQueueResult> <QueueUrl> http://sqs.us-east-1.amazon
aws.com/770098461991/queue2 </QueueUrl> </CreateQueueResult> <ResponseMetadata>
 <RequestId>cb919c0a-9bce-4afe-9b48-9bdf2412bb67</RequestId> </ResponseMetadata>
 </CreateQueueResponse>
```

错误响应的结构

如果请求不成功，则无论调用的操作如何，主要响应元素都名为 `ErrorResponse`。此元素包含 `Error` 元素和 `RequestId` 元素。每个 `Error` 都包含：

- 一个 `Type` 元素，该元素标识错误是收件人错误还是发件人错误
- 一个 `Code` 元素，该元素标识所发生错误的类型
- 一个 `Message` 元素，该元素以人类可读的格式来描述错误条件
- 一个 `Detail` 元素，该元素可能会提供有关错误的其他详细信息或者可能为空

以下是错误响应的示例。

```
<ErrorResponse> <Error> <Type> Sender </Type> <Code> InvalidParameterValue
</Code> <Message> Value (quename_nonalpha) for parameter QueueName is invalid.
 Must be an alphanumeric String of 1 to 80 in length </Message> </Error> <Re
questId> 42d59b56-7407-4c4a-be0f-4c88daeea257 </RequestId> </ErrorResponse>
```

相关主题

- 提出查询请求 (p. 25)

共享队列

Topics

Amazon SQS 包含共享您队列的方法，以便其他人可以通过使用访问控制策略中设置的权限来使用这些队列。*权限* 使其他人能够以某种特定的方式来使用您的队列。*策略* 是包含您授予的权限的实际文档。

Amazon SQS 提供了以下两种策略设置方法：简单 API 和高级 API。在简单 API 中，Amazon SQS 会为您生成访问控制策略。在高级 API 中，您可以创建访问控制策略。

共享队列的简单 API

用于共享队列的简单 API 有两个操作：

- AddPermission
- RemovePermission

通过简单 API，Amazon SQS 会基于您在 `AddPermission` 操作中包含的信息，使用所需语言编写策略。但是，Amazon SQS 生成的策略范围有限。您可以向委托人授予权限，但是无法指定限制。

共享队列的高级 API

通过高级 API，您可以自己使用访问策略语言直接编写策略，然后通过 SetQueueAttributes 操作上传策略。高级 API 允许您拒绝访问或应用更精细的访问限制（例如，基于时间或基于 IP 地址）。

如果您选择编写自己的策略，则需要了解策略的构造方式。有关策略的完整参考信息，请参阅“使用 Access Policy Language (p. 40)”。有关策略示例，请参阅“Amazon SQS 策略示例 (p. 62)”。

了解权限

权限是您向*委托人*（接收权限的用户）授予的访问类型。您可以为每个权限提供一个用于识别该权限的标签。如果您将来要删除该权限，则可以使用该标签来识别该权限。如果您要查看队列中有哪些权限，请使用 GetQueueAttributes 操作。Amazon SQS 将返回整个策略（包含所有权限）。

Amazon SQS 支持下表中所示的权限类型。

许可	描述
*	此权限类型可以向委托人授予对共享队列执行以下操作的权限：接收消息、发送消息、删除消息、更改消息的可见性、获取队列的属性。
`ReceiveMessage`	此权限类型可以授予接收队列中消息的权限。
`SendMessage`	此权限类型可以授予向队列发送消息的权限。`SendMessageBatch` 继承了与 `SendMessage` 关联的权限。
`DeleteMessage`	此权限类型可以授予从队列中删除消息的权限。`DeleteMessageBatch` 继承了与 `DeleteMessage` 关联的权限。

许可	描述
ChangeMessageVisibility	此权限类型可以授予延长或终止指定消息读取锁定超时的权限。ChangeMessageVisibilityBatch 继承了与 ChangeMessageVisibility 关联的权限。有关可见性超时的更多信息，请参阅“可见性超时 (p. 6)”。有关此权限类型的更多信息，请参阅“ChangeMessageVisibility 操作”。
GetQueueAttributes	此权限类型可以授予接收所有队列属性（只能由队列拥有者访问的策略除外）的权限。有关更多信息，请参阅“GetQueueAttributes 操作”。

Note

在为 SendMessage、DeleteMessage 或 ChangeMessageVisibility 设置权限时，系统还会为这些操作对应的批处理版本设置权限：SendMessageBatch、DeleteMessageBatch 和 ChangeMessageVisibilityBatch。不允许显式对 SendMessageBatch、DeleteMessageBatch 和 ChangeMessageVisibilityBatch 设置权限。

Amazon SQS 将各种不同权限类型的权限视为单独的权限，即使 * 包含其他权限类型提供的访问权限也是如此。例如，可以向用户同时授予 * 和 SendMessage 权限，即使 * 包含 SendMessage 提供的访问权限也是如此。

此概念在您删除权限时适用。如果委托人只有 * 权限，则请求删除 SendMessage 权限不会为委托人留下“除此以外的一切”权限。相反，该请求不会执行任何操作，因为委托人之前不具有显式 SendMessage 权限。

如果您要删除 * 并且只为委托人保留 ReceiveMessage 权限，请先添加 ReceiveMessage 权限，然后再删除 * 权限。

Tip

您可以为每个权限提供一个用于识别该权限的标签。如果您将来要删除该权限，则可以使用该标签来识别该权限。

Note

如果您要查看队列中有哪些权限，请使用 GetQueueAttributes 操作。将返回整个策略（包含所有权限）。

授予对队列的匿名访问权限

您可以向匿名用户授予共享队列访问权限。此类访问权限无需任何签名或访问密钥 ID。

要允许匿名访问，您必须编写自己的策略，并将 Principal 设置为 *。有关编写您自己的策略的信息，请参阅“使用 Access Policy Language (p. 40)”。

Caution

请记住，队列拥有者负责承担与队列相关的所有费用。因此，您可能希望以其他某种方式（例如，按时间或 IP 地址）限制匿名访问。

编程语言

AWS 为倾向于使用语言特有的 API 而非 Amazon SQS 的 Query API 来构建应用程序的软件开发者提供库文件、示例代码、教程和其他资源。这些库文件提供基本功能（不包括在 Amazon SQS 的 Query API 中），例如请求身份验证、请求重试和错误处理，以便您更轻松地开始使用。库文件和资源可供下列语言使用：

- Java
- PHP
- Python
- Ruby
- Windows 和 .NET

有关以所有语言推出的库和示例代码，请转到 Sample Code & Libraries。

有关移动应用程序开发，请参阅：

- AWS SDK for Android
- AWS SDK for iOS

使用 Access Policy Language

Topics

本部分内容面向希望编写自己的访问控制策略的 Amazon SQS 用户。如果您希望仅基于 AWS 账户 ID 和基本权限（例如，SendMessage、ReceiveMessage）来允许访问，则不需要编写自己的策略。在这种情况下，您可以仅使用 Amazon SQS `AddPermission` 操作。如果您希望基于更精细的条件（例如，请求到达的时间或请求者的 IP 地址）来显式拒绝访问或允许访问，则需要编写自己的策略并使用 Amazon SQS `SetQueueAttributes` 操作将其上传到 AWS 系统。

Note

要编写您自己的策略，您必须熟悉 JSON。有关更多信息，请转到 http://json.org。

本节的主要部分包括您需要了解的基本概念、编写策略的方法，以及 AWS 用来评估策略并决定是否向请求者授予资源访问权限的逻辑。虽然本部分中的大多数信息都是服务兼容的，但是其中也有一些您需要了解的特定于 SQS 的详细信息。有关更多信息，请参阅 Amazon SQS 策略的特别信息 (p. 64)。

概述

Topics

本部分介绍了要使用access policy language编写策略需要了解的基本概念。本部分还介绍了访问控制与access policy language一起使用的一般过程以及如何评估策略。

何时使用访问控制

对于如何授权或拒绝资源访问，您有很大的灵活性。然而，普通的使用案例都相当简单。

- 您欲授予另一 AWS 账户特定访问类型，以访问您的队列（例如 SendMessage）。有关更多信息，请参阅 使用案例 1 (p. 49)。
- 您欲授权另一 AWS 账户可在指定时段内访问您的队列。有关更多信息，请参阅 使用案例 2 (p. 49)。
- 您欲授权另一 AWS 账户仅在从 Amazon EC2 实例发出请求的情况下访问您的队列。有关更多信息，请参阅 使用案例 3 (p. 49)。
- 您欲*拒绝* 另一 AWS 账户访问您的队列。有关更多信息，请参阅 使用案例 4 (p. 50)。

主要概念

以下几部分介绍了要使用access policy language需要了解的概念。它们都按逻辑顺序介绍，您需要了解的第一项术语在清单最前面。

许可

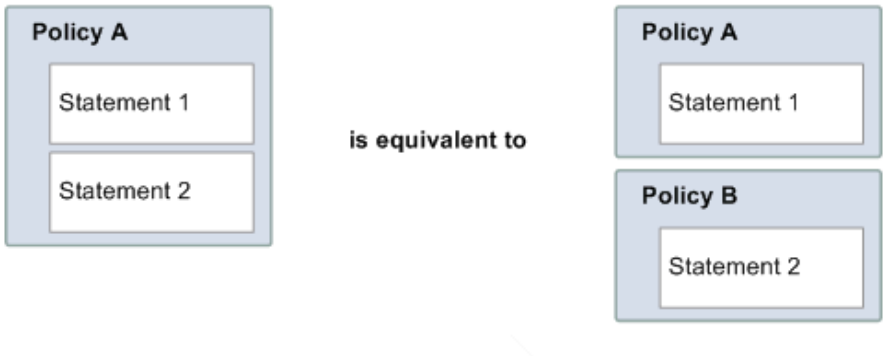
权限 是指允许或不允许对某一特定资源进行某种类型的访问的概念。权限基本遵循以下形式："在 D 适用的情况下，A 被允许或不被允许对 C 执行 B"。例如，只要 *Jane* (A) *在 2009 年 5 月 30 日午夜之前请求接收消息* (D)，她就有权从 *John 的 Amazon SQS 队列* (C) *接收消息* (B)。无论 Jane 何时向 Amazon SQS 发送使用 John 的队列的请求，该产品都会检查她是否拥有权限，同时还会检查发送的请求是否满足 John 在权限中设定的条件。

语句

语句 是指对用access policy language编写的单个权限的正式说明。您通常编写的语句是一个更大的容器式文档，被称为*策略*（见下一概念），的一部分。

策略

策略 是一个使用access policy language编写的文档，用作存放一个或多个语句的容器。例如，策略中可包含两个语句：一个语句声明 Jane 可以使用 John 的队列，另一个语句声明 Bob 不得使用 John 的队列。如下图所示，一个等效情境中将包含两份策略，一份包含 Jane 可使用 John 的队列的声明，另一份包含 Bob 不得使用 John 的队列的声明。



AWS 产品使用声明（无论是包含在单一策略，或是多份策略中）中的信息实施访问控制（例如 Amazon SQS），以确定是否授权请求访问资源者访问资源。我们经常将*策略*与*语句*这两个词互换，因为二者一般表示相同的概念（代表权限的实体）。

发布者

*发布者*是指编写策略以授予资源权限的人。发布者（按定义）通常指资源所有者。AWS 不允许 AWS 服务用户为他们不拥有的资源创建策略。如果 John 是资源所有者，那么当他提交他编写的策略为该资源授予权限时，AWS 会对 John 的身份进行认证。

委托人

*委托人*是指在策略中获得权限的一个或多个人。委托人是“如果 D 适用的情况下，那么 A 可以对 C 执行 B ”语句中的 A。在一个策略中，您可将委托人设置为“任何人”（例如，如您可指定一个通配符代表所有人。您这样操作，例如，如果您不想根据请求者的实际身份限制访问，那么您可以根据其他的识别特征比如请求者的 IP 地址。

操作

*操作*是指委托人有权执行的活动。操作是“如果 D 适用的情况下，那么 A 可以对 C 执行 B”语句中的 B。通常情况下，该操作只是向 AWS 提出请求的操作。例如，Jane 使用 `Action=ReceiveMessage` 向 Amazon SQS 发送请求。在一个策略中您可指定一个或多个操作。

资源

*资源*是指委托人请求访问的数据元。在表述“在满足 D 的情况下，A 拥有对 C 执行 B 的许可”中，C 即指资源。

条件与密钥

*条件*是指有关权限的任何限制和详细信息。条件是“如果 D 适用的情况下，那么 A 可以对 C 执行 B”语句中的 D。说明条件的策略部分可能是整个部分最详细且最复杂的内容。普通条件与以下项目相关：

- 日期和时间（例如：请求必须在指定日期前到达）
- IP 地址（例如，请求者的 IP 地址必须是某个特定 CIDR 范围的一部分）

*密钥*是指作为访问限制的基础的特定特性。例如，访问日期和时间。

您同时使用*条件*和*密钥*来明确说明限制。通过最简易的方法，举例说明某一个限制是如何真正被实现的：如果您想在 2010年 5 月 30 日前限制访问，则可以使用名为 `DateLessThan` 的条件。可以使用名为 `AWS:CurrentTime` 的密钥并将其值设置为 `2010-05-30T00:00:00Z`。AWS 确定您能使用的条件和密

钥。此外，AWS 产品本身（例如，Amazon SQS）也可能会定义特定于服务的密钥。有关条件的更多信息，请参阅“条件 (p. 55)”。有关可用密钥的更多信息，请参阅“可用密钥 (p. 57)”。

请求者

请求者 是指向 AWS 产品发送请求并要求访问某一特定资源的人。请求者向 AWS 发送请求，基本都会说：“在 D 适用的情况下，您能让我对 C 执行 B 吗？”

评估

评估 是指AWS产品根据适用的策略决定应拒绝还是允许传入请求的过程。有关评估逻辑的信息，请参阅“评估逻辑 (p. 46)”。

效果

效果 是指您希望策略语句在评估时返回的结果。当您在策略中编写语句时需指定此值，可能的值包括 *deny* 和 *allow*。

例如，您可以编写一个策略，其中包含*拒绝* 所有来自南极洲地区的请求（效果=如果请求使用分配给南极洲的 IP 地址，则拒绝）的语句。同样地，您可以编写一个策略，其中包含*允许* 所有*并非* 来自南极洲地区的请求（效果=如果请求并非来自南极洲地区，则允许）。虽然这两个语句看似执行相同的操作，但是在access policy language逻辑上，它们是不同的。有关更多信息，请参阅 评估逻辑 (p. 46)。

虽然您只能为效果指定两种可能的值（允许或拒绝），但是在策略评估时可能会有三种不同的结果：*默认拒绝*、*允许* 或*显式拒绝*。有关更多信息，请参阅以下概念以及“评估逻辑 (p. 46)”。

默认拒绝

默认拒绝 是没有允许或显式拒绝的策略中的默认结果。

允许

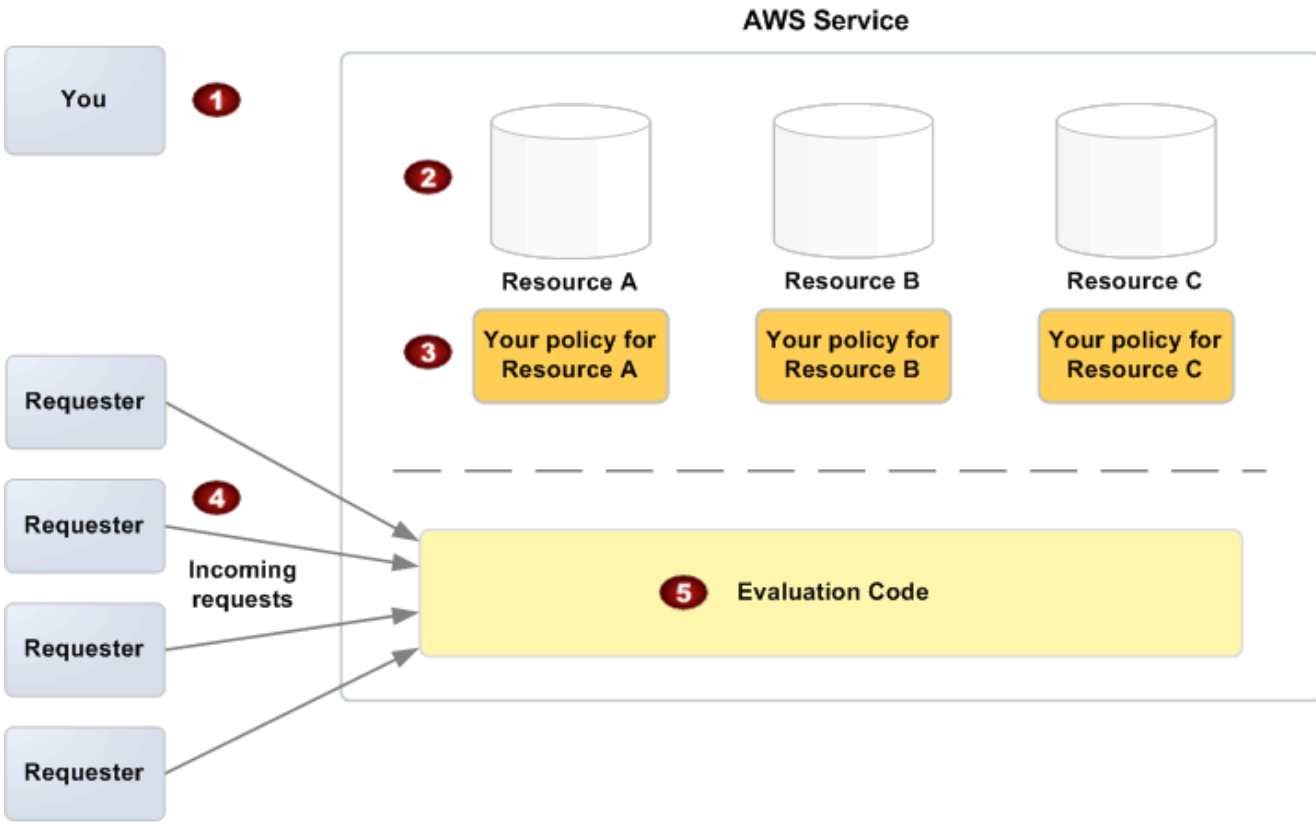
假设任何声明的条件均已满足，效果=允许的语句会得到*允许* 结果。示例:如果请求是在 2010 年 4 月 30 日下午 1 点接收的，则允许这些请求。允许将覆盖所有的默认拒绝，但绝不能覆盖显式拒绝。

显式拒绝

假设任何声明的条件均已满足，效果=拒绝的语句会得到*显式拒绝* 结果。示例:如果请求来自南极洲地区，则全部拒绝。不管其他什么策略会允许，所有来自南极洲地区的请求都会被拒绝。

架构概述

下列图和表格介绍了为您提供资源访问控制的相互作用的主要组成部分。

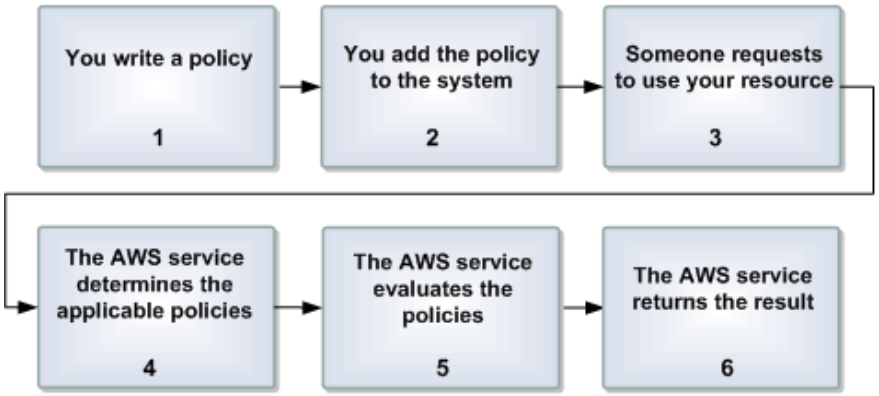


1	您，资源所有者。
2	您的资源（包含在 AWS 产品内，例如，Amazon SQS 队列）。
3	您的策略。 通常情况下，每个资源拥有一个策略，虽然您可以有多个。ASW 服务本身提供一个您用来上传和管理您的策略 API。有关策略内容的信息，请参阅“如何编写一个策略。 (p. 51)”。
4	请求者和他们向 AWS 服务传入的请求。
5	access policy language评估代码。 这是一组在 AWS 服务内能根据适用的策略对传入的请求进行评估并决定是否允许该请求者访问资源的代码。有关此服务如何做出决定的信息，请参阅“评估逻辑 (p. 46)”。

有关各组件如何协作的一般流程，请参阅“使用Access Policy Language (p. 45)”。

使用Access Policy Language

下列图表介绍了访问控制与access policy language协作的一般过程。



将访问控制与Access Policy Language一起使用的过程

1	为您的资源编写一个策略。 例如，您编写策略来为 Amazon SQS 队列指定权限。有关更多信息，请参阅 如何编写一个策略。 (p. 51)。
2	上传您的策略至 AWS。 AWS 服务自身提供一个您用来上传您的策略的 API。例如，您使用 Amazon SQS `SetQueueAttributes` 操作来为特定的 Amazon SQS 队列上传策略。
3	某人向您发出使用您的资源的请求。 例如，某用户向 Amazon SQS 发送使用您的一个队列的请求。
4	AWS 服务决定哪些策略能适用于该请求。 例如，Amazon SQS 将查看所有可用的 Amazon SQS 策略，并确定哪些策略适用（基于资源是什么、请求者是谁等）。
5	AWS 服务将对这些策略进行评估。 例如，Amazon SQS 将对策略进行评估，确定是否允许请求者使用您的队列。有关决策逻辑的信息，请参阅“评估逻辑 (p. 46)”。
6	AWS 服务或许会拒绝请求，或许会继续处理这个请求。 例如，根据策略评估结果，服务将返回一个“访问被拒绝”的错误信息给请求者，或继续处理该请求。

相关主题

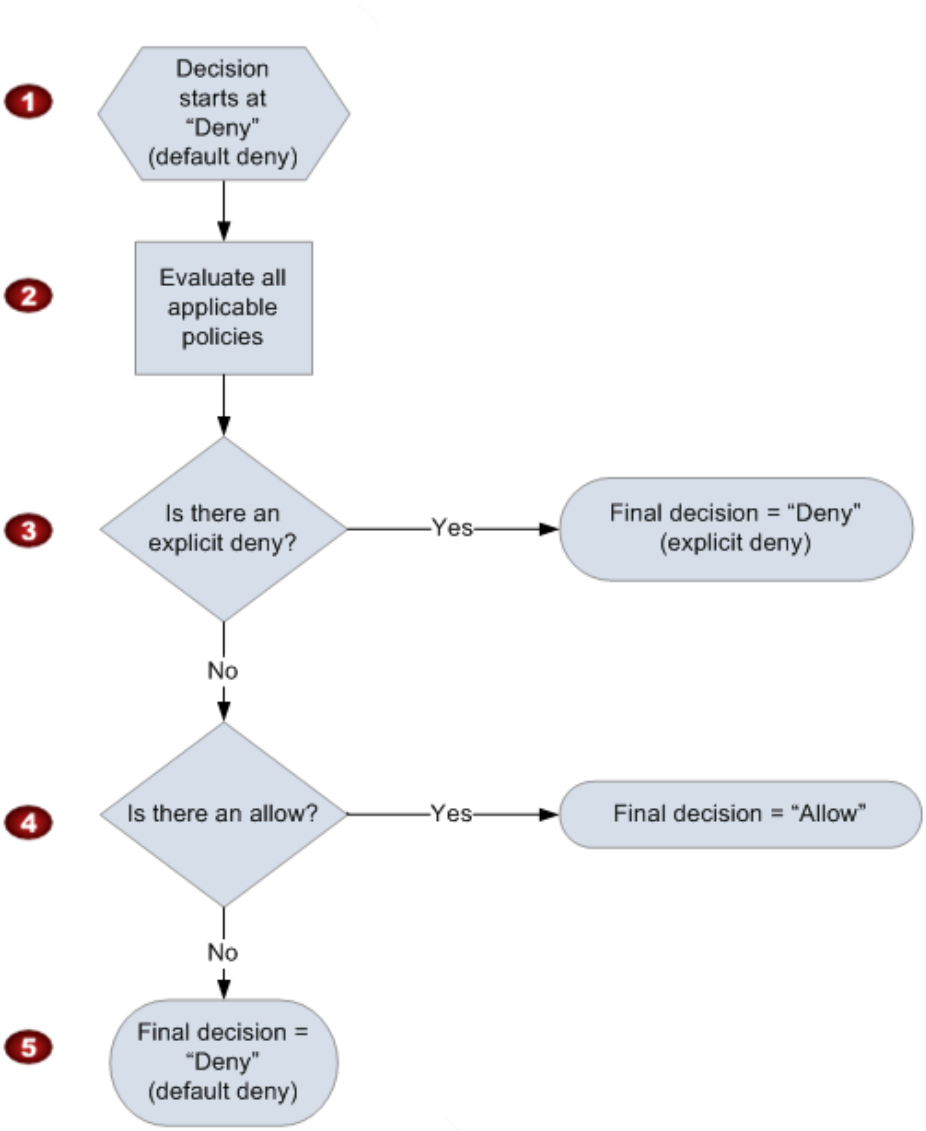
- 架构概述 (p. 43)

评估逻辑

评估时的目标是决定应允许还是拒绝除您（资源拥有者）以外的某个人发送的给定请求。评估逻辑遵循多个基本规则：

- 在默认情况下，除了您，任何人提出使用您资源的请求均会被拒绝。
- 一个允许可以超控任何其他默认拒绝
- 一个显式拒绝可以超控任何允许
- 策略评估的顺序不重要

以下流程图和讨论更加详细地描述了如何做出决定。



1	决定开始是一个默认拒绝。

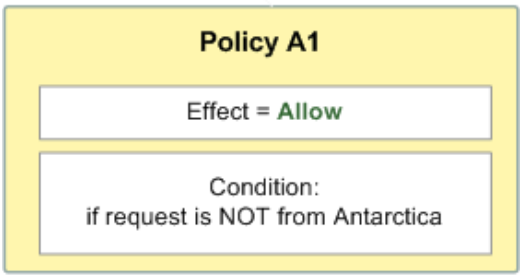
2	然后执行代码将评估适用于请求的所有策略（根据资源、委托人、操作和条件）。 执行代码评估策略的顺序不重要。
3	在所有这些策略中，执行代码将寻找一个能适用于请求的显式拒绝指令。 只要找到一个显式拒绝，执行代码就会返回一个“拒绝”决定，过程将完成（这是一个显式拒绝；有关更多信息，请参阅“显式拒绝 (p. 43)”）。
4	如果没有找到显式拒绝，那么执行代码将寻找适用于请求的任何“允许”指令。 如果它还是找到了一个，那么执行代码将返回一个“允许”决定，且整个过程完成（服务将继续处理该请求）。
5	如果没有找到允许，那么最终的决定将是“拒绝”（因为没有显式拒绝或允许，所以这将被视为一个*默认拒绝*（有关更多信息，请参阅“默认拒绝 (p. 43)”）。

显式拒绝和默认拒绝的相互作用

如果策略不直接适用于请求，那么策略将产生一个默认拒绝。例如，如果用户请求使用 Amazon SQS，但适用于该用户的唯一策略表示该用户可使用 Amazon SimpleDB，则该策略将导致“默认拒绝”的结果。

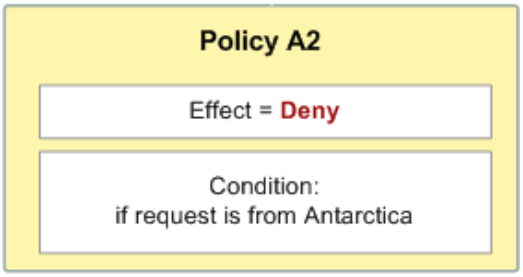
如果一个语句中的某个条件未被满足，那么策略将产生一个默认拒绝。如果声明中的所有条件都满足，那么根据策略中的效果元素的值，策略或许会产生允许，或许会产生显式拒绝。如果一个条件未被满足，策略没有指定如何处理，那么在那种情况下默认值将产生一个默认拒绝。

例如，假设您想要阻止来自南极洲地区的请求进入。只要请求不是来自于南极洲地区，您编写的策略（称作策略 A1)将允许接受请求。下列示意图说明了该策略。



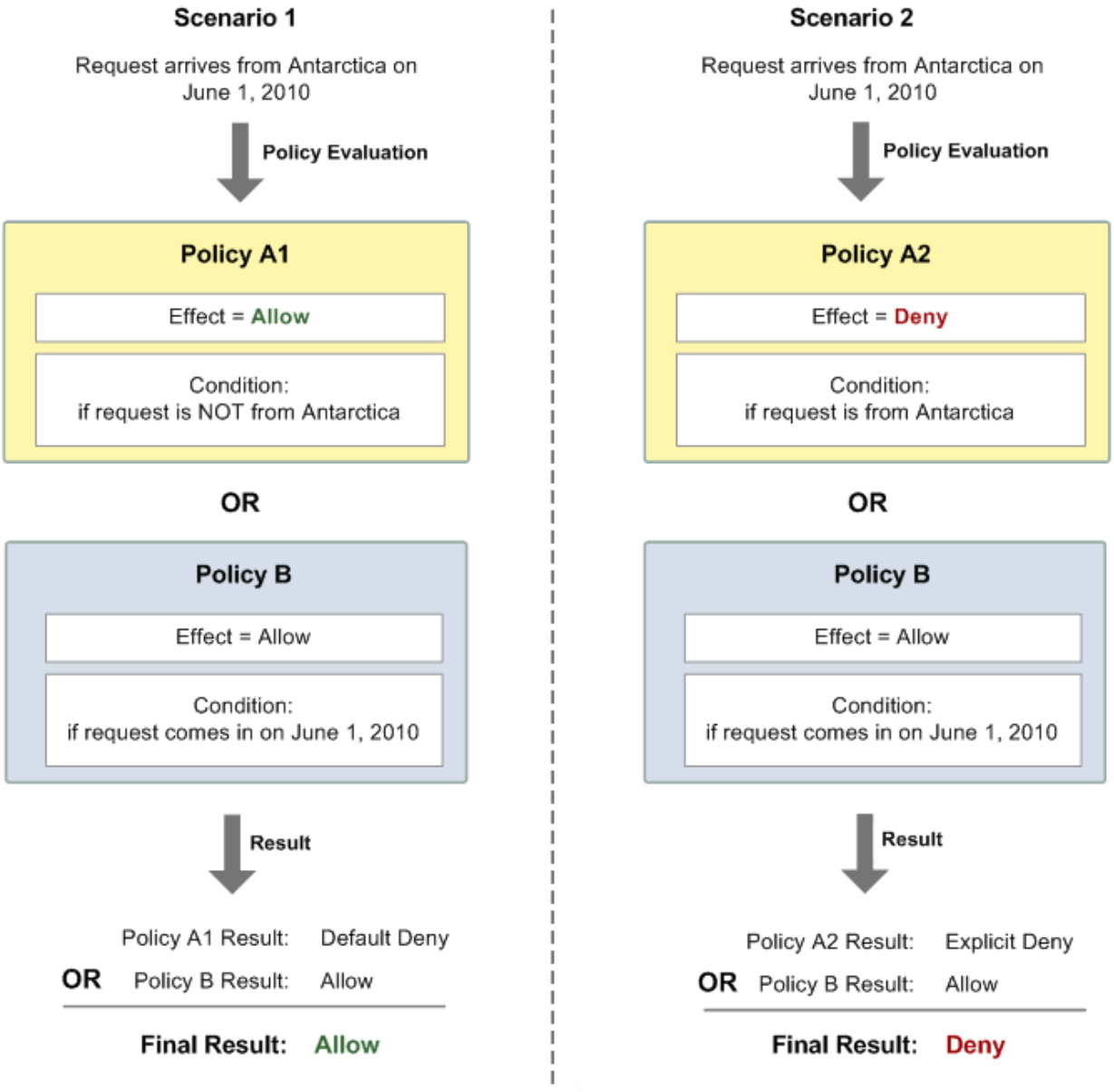
如果某人从美国发出请求，那么条件已经满足（该请求不是来自南极洲）。因此，该请求将被允许。但是，如果某人从南极洲地区发出请求，那么条件未满足，因此策略结果将是默认拒绝。

您可通过按照下列示意图重新编写策略（称作策略 A2）将结果转变为一个显式拒绝。此时，如果请求是来自南极洲地区，那么策略将明确拒绝该请求。



如果某人从南极洲发出请求，那么条件已经满足，策略的结果将是一个显式拒绝。

默认拒绝和显式拒绝的区别很重要，因为默认拒绝可以被允许覆盖，但是显式拒绝就不能。例如，比如说如果请求在 2010 年 6 月 1 日到达，那么将有另一个策略允许这些请求。当与限制来自南极洲的访问的策略相结合时，这个策略将如何影响综合结果呢？当将按日期要求设置的策略与上述策略 A1 和 A2 相结合时，我们将对比综合结果。方案 1 是将策略 A1 与策略 B 相结合，方案 2 是将策略 A2 与策略 B 相结合。以下图表和讨论显示了如果于 2010 年 6 月 1 日从南极洲地区发出请求输入时的结果。



在方案 1 中，策略 A1 将返回一个默认拒绝，如本节之前所描述的那样。因为该策略（按照定义）允许 2010 年 6 月 1 日发出的请求，所以策略 B 将返回允许。从策略 B 发出的允许覆盖了从策略 A1 发出的默认拒绝，因此该请求会被允许。

在方案 2 中，策略 B2 返回了一个显式拒绝，如本节之前所描述的那样。此外，策略 B 返回了一个允许。从策略 A2 发出的显式拒绝将超控从策略 B 发出的允许，因此该请求会被拒绝。

关于访问控制的基本使用案例

本节介绍了几个访问控制典型使用案例。

使用案例 1

假设您在 Amazon SQS 系统中拥有一个队列集。在最简单的情况下，您可能希望授予一个或多个 AWS 账户特定访问类型，以访问某一队列（例如 SendMessage、ReceiveMessage）。

通过 Amazon SQS API 操作 `AddPermission` 即可轻松做到这一点。此操作需要几个输入参数，并且会在 Amazon SQS 系统中自动针对该队列创建一个策略。对于该使用案例而言，因为 Amazon SQS 可为您自动创建策略，因此，您无需阅读本附录，亦无需了解如何自行编写策略。

以下示例显示了授予 AWS 账户 ID 1111-2222-3333 从您名为 queue2 的队列发送和接收请求的权限的策略。在本例中，您的 AWS 账户 ID 为 4444-5555-6666。

```
{ "Version":"2012-11-05", "Id":"UseCase1", "Statement" : [ { "Sid":"1", "Ef
fect":"Allow", "Principal" : { "AWS": "111122223333" }, "Action":["sqs:SendMes
sage","sqs:ReceiveMessage"], "Resource": "arn:aws:sqs:us-east-
1:444455556666:queue2", } ] }
```

使用案例 2

在这一使用案例中，您希望*仅在指定时段内*允许一个或多个 AWS 账户访问您的队列。

您需要了解如何针对队列自行编写策略，因为当授权某人访问您的队列时，Amazon SQS `AddPermission` 操作禁止您执行指定时限的操作。在这一案例中，您需要自行编写策略，然后利用 `SetQueueAttributes` 操作将其上传至 AWS 系统。该操作会有效地将您的策略设置为队列属性。

下列示例与使用案例 1 相同，但有一点不同的是其包括一个“在 2009 年 6 月 30 日中午之前 (UTC) 限制访问”的条件。

```
{ "Version":"2012-11-05", "Id":"UseCase2", "Statement" : [ { "Sid":"1", "Ef
fect":"Allow", "Principal" : { "AWS": "111122223333" }, "Action":["sqs:SendMes
sage","sqs:ReceiveMessage"], "Resource": "arn:aws:sqs:us-east-
1:444455556666:queue2", "Condition" : { "DateLessThan" : { "AWS:Current
Time":"2009-06-30T12:00Z" } } } ] }
```

使用案例 3

在这一使用案例中，您希望*仅在由 Amazon EC2 实例发出请求时*允许访问您的队列。

再次重申，您需要了解如何自行编写策略，因为当授权访问您的队列时，Amazon SQS `AddPermission` 操作禁止您执行指定 IP 地址限制的操作。

下列示例以使用案例 2 中的示例为基础，同时还包括一个“限制 IP 范围 10.52.176.0/24 的访问”的条件。因此，在此示例中，AWS 账户 1234-5678-9012 只有在 2009 年 6 月 30 日中午之前从地址范围为 10.52.176.0/24 的 IP 地址发出向或从 queue2 发送或接收消息的请求，请求才能得到允许。

```
{ "Version":"2012-11-05", "Id":"UseCase3", "Statement" : [ { "Sid":"1", "Ef
fect":"Allow", "Principal" : { "AWS": "111122223333" }, "Action":["sqs:SendMes
sage","sqs:ReceiveMessage"], "Resource": "arn:aws:sqs:us-east-
1:444455556666:queue2", "Condition" : { "DateLessThan" : { "AWS:Current
```

```
Time":"2009-06-30T12:00Z" }, "IpAddress" : { "AWS:SourceIp":"10.52.176.0/24" }
 } } ] }
```

使用案例 4

在这一使用案例中，您希望*拒绝* 某一特定 AWS 账户访问您的队列。

再次重申，您需要了解如何自行编写策略，因为 Amazon SQS `AddPermission` 操作禁止您执行*拒绝* 访问队列的操作，而您只能*授予* 访问权限。

下列示例与第一个使用案例 (#1) 相同，但有一点不同的是其*拒绝* 访问指定的 AWS 账户。

```
{ "Version":"2012-11-05", "Id":"UseCase4", "Statement" : [ { "Sid":"1", "Ef
fect":"Deny", "Principal" : { "AWS": "111122223333" }, "Action":["sqs:SendMes
sage","sqs:ReceiveMessage"], "Resource": "arn:aws:sqs:us-east-
1:444455556666:queue2", } ] }
```

从这些使用案例中，您会发现当您希望执行基于特别条件的限制访问或完全拒绝某人的访问时，需要阅读本附录，并了解如何自行编写策略。您还会发现，策略本身并不复杂，访问策略语言也很简单直接。

如何编写一个策略。

Topics

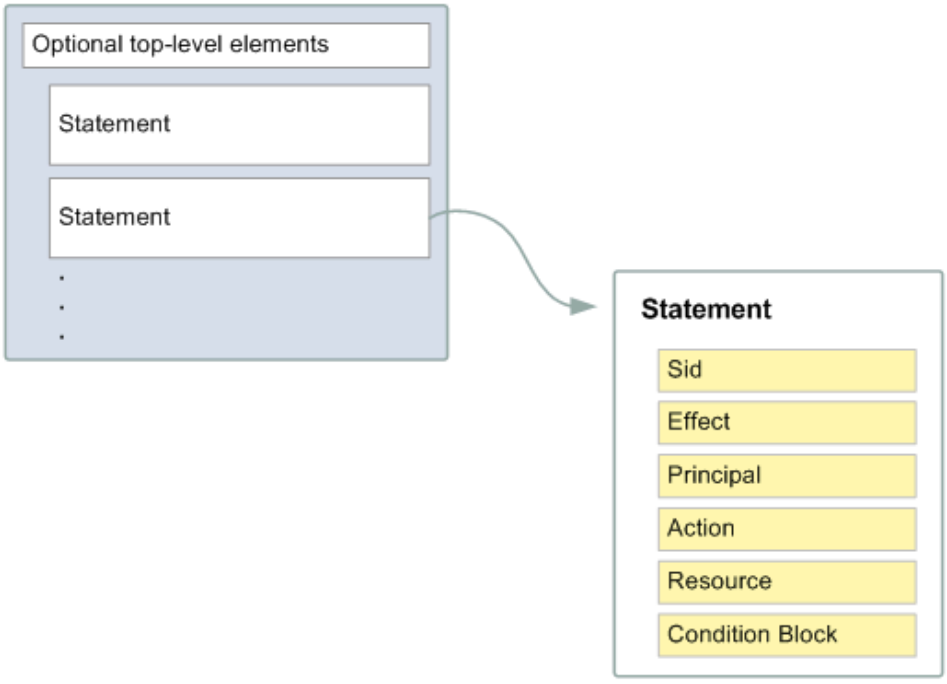
本节介绍了如何编写策略并为有关每个策略元素提供了参考信息。

基本策略结构

每个策略是一个 JSON 格式文件。如下图所示，一个策略包含：

- 可选的策略广泛信息（位于文件顶部）
- 一个或多个单独的*语句*

每个语句都是有关单独权限的核心信息。如果一个策略包含多个语句，那么在评估时我们将跨这些语句应用一个 OR 逻辑值。如果多个策略都适用于一个请求，那么在评估时我们将跨这些策略应用一个 OR 逻辑值。



语句中的信息均包含在一系列*元素* 内。有关这些元素的信息，请参阅“元素描述 (p. 52)”。

Example

如下列这一简单的策略所述，AWS 开发人员可在美国东部（弗吉尼亚北部）地区中使用账户 ID 111122223333 向 Amazon SQS 队列（由账户 ID 为 444455556666 的开发人员所有）发送和读取请求，但条件是必须在 2009 年 6 月 30 日中午 (UTC) 之前从地址范围 10.52.176.0/24 发出请求。

```
{ "Version":"2012-10-17", "Id":"cd3ad3d9-2776-4ef1-a904-4c229d1642ee", "State
ment" : [ { "Sid":"1", "Effect":"Allow", "Principal" : { "aws": "111122223333"
 }, "Action":["sqs:SendMessage","sqs:ReceiveMessage"], "Resource":
"arn:aws:sqs:us-east-1:444455556666:queue2", "Condition" : { "IpAddress" : {
"aws:SourceIp":"10.52.176.0/24" }, "DateLessThan" : { "aws:CurrentTime":"2009-
06-30T12:00Z" } } } ] }
```

元素描述

Topics

此部分描述的是在一个策略及其语句中的可用元素。这些元素按照在策略中使用的大致顺序列出。`Id`、`Version` 和 `Statement` 是顶级策略元素；其余元素是语句级元素。提供了几个 JSON 例子。

所有元素都可供选择来对策略文件本身进行解析。元素的顺序无关紧要（例如，`Resource` 元素可以出现在 `Action` 元素之前）。不必指定策略中的任何条件。

版本

`Version` 元素指定access policy language的版本。仅允许以下值：

- `2012-10-17`.这是策略语言的当前版本，您应将此版本号用于所有策略。
- `2008-10-17`.这是策略语言的早期版本。您可能会在现有策略中看到此版本。请勿将此版本用于任何新策略或正在更新的任何现有策略。

如果您不包括 `Version` 元素，则值默认为 `2008-10-17`。但是，较好的做法是始终包括 `Version` 元素并将其设置为 `2012-10-17`。

Note

如果您的策略包括*策略变量*，则您*必须* 包括 `Version` 元素并将其设置为 `2012-10-17`。如果您不包括设置为 `2012-10-17` 的 `Version` 元素，则系统不会将 `${aws:username}` 之类的变量视为变量，而会将其视为策略中的文本字符串。有关更多信息，请参阅《*使用 IAM* 指南》中的“策略变量”部分。

```
"Version":"2012-10-17"
```

Id

Id 元素指定策略的可选标识符。不同产品使用的 ID 也不尽相同。

对于 IAM 策略而言，该产品可将策略的 Id 值自动设置为您创建的策略名称。若您试图 设置 Id 元素，则策略会被拒绝。IAM 策略的 Id 值看上去可能与下例类似。

```
"Id":"Admin_Policy"
```

对于允许您设置 ID 元素的产品，我们建议您使用 UUID (GUID) 值，或将 UUID 合并为 ID 的一部分，以确保唯一性。

```
"Id":"cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

Note

某些 AWS 产品（例如，Amazon SQS 或 Amazon SNS）可能需要此元素并要求其具有唯一性。有关编写策略的特定于服务的信息，请参阅您使用的服务的文档。

语句

Statement 是语句的主要元素。可以包括多个元素（参见本指南的后面部分）。

Statement 元素包括一组独立的语句。每一个独立语句表示波形括号内{ }的一个不同的 JSON 块。

```
"Statement":[{...},{...},{...}]
```

Sid

Sid（语句 ID）是您为策略语句提供的一个可选标识符。本质上它只是策略文件 ID 的一个子 ID。

Important

实施access policy language的 AWS 产品（例如，Amazon SQS 或 Amazon SNS）可能需要此元素并要求其具有唯一性。有关编写策略相关的特定于服务的信息，请参阅“Amazon SQS 策略的特别信息 (p. 64)”。

```
"Sid" : "1"
```

效果

Effect 是一个必需的元素，用来表示您希望该语句的结果为允许还是显式拒绝（有关更多信息，请参阅“显式拒绝 (p. 43)”）。

Effect 的有效值为 Allow 和 Deny。

```
"Effect":"Allow"
```

委托人

Principal 是指根据策略接受权限或者权限被拒绝的一个或多个用户。您必须通过委托人的 AWS 账户 ID 来指定委托人（例如 1234-5678-9012，包括或不包括连字符）。您可以指定多个委托人，或者使用通配符 (*) 来表示所有可能的用户。您可以访问 http://aws.amazon.comhttp://aws.amazon.com 登录您的 AWS 账户，然后单击账户活动来查看您的账户 ID。

在 JSON 格式下，您可以使用 "AWS": 作为委托人 AWS 账户 ID 的前缀。在下列样例中，语句中包含两个委托人。

```
"Principal":{ "AWS":["123456789012", "999999999999"]}
```

非委托人

若要破例处理一列委托人，可以使用 NotPrincipal 元素。例如，如果您要阻止某个特定账户以外的所有 AWS 账户，则可能会使用此元素。

在 JSON 格式下，您可以使用 "AWS": 作为委托人 AWS 账户 ID 的前缀。在以下示例中，语句包括一个委托人，该委托人会由于 NotPrincipal 元素的原因而排除在委托人列表之外。

```
"NotPrincipal":{ "AWS":["123456789012"]}
```

操作

Action 是指允许或拒绝的特定访问类型（例如，读取或写入）。您可针对该元素指定多个值。这些值的格式自由，但必须与 AWS 产品的预期值相匹配（有关更多信息，请参阅“Amazon SQS 策略的特别信息 (p. 64)”）。可以使用通配符(*)授予委托人访问权限，进行特定 AWS 服务所允许与其他开发商共享的全部操作。例如，Amazon SQS 允许您仅共享所有可能的 Amazon SQS 操作的一个特定子集。因此，使用通配符无法使用户完全控制队列，它只能提供对操作的特殊子集的访问权限。

```
"Action":["sqs:SendMessage","sqs:ReceiveMessage"]
```

前缀和操作名称不区分大小写。例如，sqs:SendMessage 等同于 SQS:sendmessage。

非操作

若要破例处理一列操作，则可以使用 NotAction 元素。例如，如果您希望您的用户只能使用 Amazon SQS SendMessage，则可能会使用此元素。

以下示例引用了除 Amazon SQS SendMessage 以外的所有操作。为阻止用户访问任何其他操作，您在策略中将用到该元素，格式为 "Effect":"Deny"。

除指定操作外，NotAction 元素与所有操作匹配。若要破例处理一列允许或拒绝的操作，则可以使用此元素。下面的示例与 Publish 之外的任何操作匹配。

```
"NotAction":"sqs:SendMessage"
```

资源

Resource 是指策略包含的单个或多个数据元。此值可在字符串中任何位置包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。这些值的格式自由，但必须遵循 AWS 服务的预期格式。例如，对于 Amazon SQS，您可以使用以下格式指定队列：<account ID of queue owner>:<queue name>.例如：111122223333:queue1.

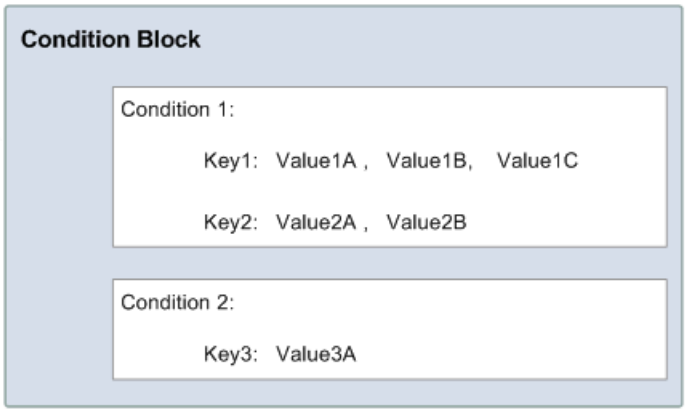
```
"Resource":"arn:aws:sqs:us-east-1:111122223333:queue1"
```

条件

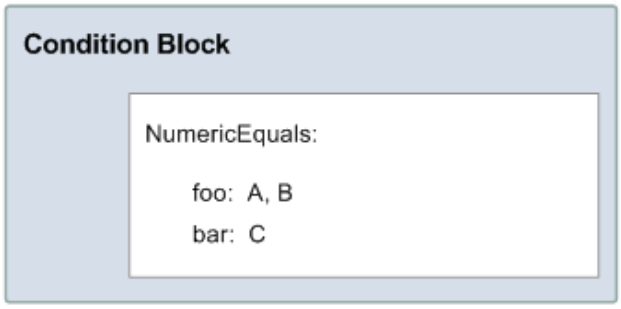
此部分介绍了 Condition 元素以及可在此元素内使用的信息。

条件块

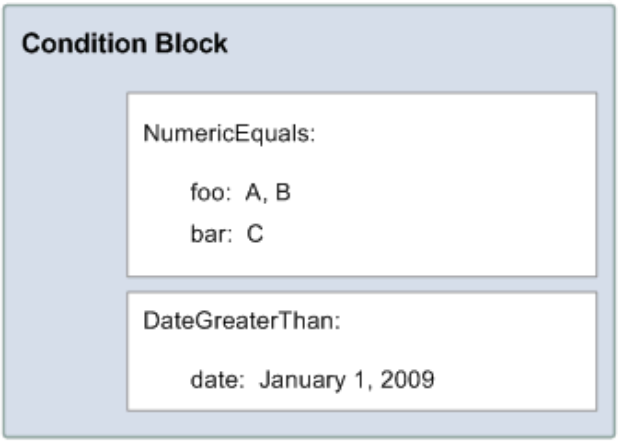
Condition 元素是策略语句中最复杂的部分。我们将其称为*条件块*，原因在于它虽然是单个 Condition 元素，但却包含了多个条件，并且每个条件都包括多个键值对。下图对此进行了说明。除非对个别密匙有特别说明，所有的密匙都可以有多个值。



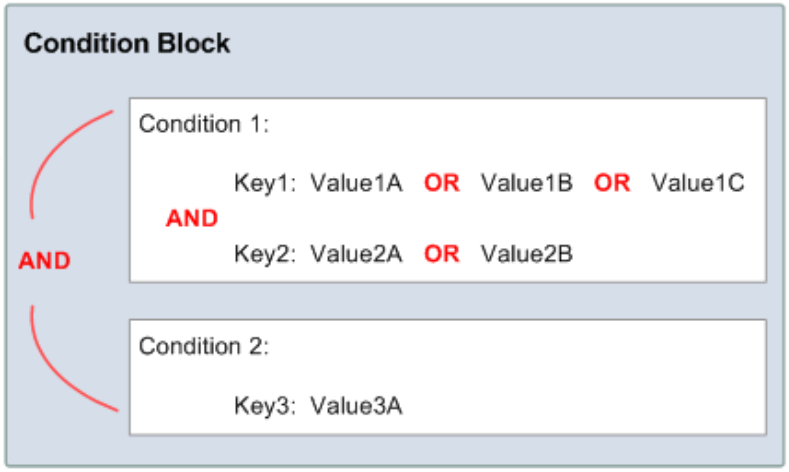
创建条件块时，您需对每个条件命名，给每个条件指定至少一个键值对。AWS 规定了可以使用的条件和密匙（在后面部分列出）。条件示例包括 NumericEquals。假如您有一个虚构的资源，并希望仅在某个特定的数值 *foo* 等于 A 或 B，且另一个数值 *bar* 等于 C 时，John 才能使用这个资源。那么您需要创建如下图所示的条件块。



假如您还想要限制 John 在 2009 年 1 月 1 日之后的访问，则需要另外增加一个日期为 2009 年 1 月 1 日的条件，DateGreaterThan。这样这个条件块看上去就跟下图一样了。



如下图所示，我们通常给条件块内的条件和条件内的密钥赋一个 AND 逻辑值。对于单个密钥，我们通常给这些值赋一个逻辑 OR 值。所有的条件都必须满足以返回到允许或明确的拒绝决定。如果一个条件没有满足，那么结果为默认拒绝。



如上所述，AWS 定义了您可以使用的条件和密钥（例如其中一个密钥是 `aws:CurrentTime`，让您可以根据日期和时间来限制访问）。AWS 服务本身同样可以规定自身的特定于服务的密钥。有关可用密钥的列表，请参阅“可用密钥 (p. 57)”。

对于使用真实密钥的具体示例，假设您希望 John 可以在以下三个条件下访问您的 Amazon SQS 队列：

- 时间为 2010 年 8 月 16 日中午 12 点之后
- 时间为 2010 年 8 月 16 日下午 3 点之前
- 发送请求的 IP 地址为 192.168.176.0/24 或 192.168.143.0/24 范围之内

您的条件块有三个单独的条件，并且所有的三个条件都必须满足才允许 John 访问您的队列。

条件块在你们策略中的样子如下所示。

```
"Condition" : { "DateGreaterThan" : { "aws:CurrentTime" : "2009-04-16T12:00:00Z"
 }, "DateLessThan": { "aws:CurrentTime" : "2009-04-16T15:00:00Z" }, "IpAddress"
 : { "aws:SourceIp" : ["192.168.176.0/24","192.168.143.0/24"] } }
```

可用密钥

AWS 提供了一组受所有 AWS 产品支持并适合access policy language的公用密钥来进行访问控制。这些密钥是：

- `aws:CurrentTime`–适用于日期/时间条件（请参阅“日期条件 (p. 58)”）
- `aws:EpochTime`–用纪元或 UINX 时间表示的日期，用于日期/时间条件（请参阅“日期条件 (p. 58)”）
- `aws:MultiFactorAuthAge`–密钥可以提供一个数值，这个数值能显示通过多重验证 (MFA) 的安全证书发出的请求信息多久发出（以秒为单位）。与其他密钥不同，如果多重验证 (MFA) 使用不成功，那么这个密钥就不会出现（请参阅“条件密钥的存在 (p. 60)”、“数字条件 (p. 58)”和“将多重验证 (MFA) 设备与 AWS 结合使用）。
- `aws:principaltype`–检查当前请求的委托人类型（用户、账户、联盟用户等）（请参阅“字符串条件 (p. 57)”）。
- `aws:SecureTransport`–表示请求是否使用 SSL 发送的布尔值（请参阅“布尔值条件 (p. 59)”）
- `aws:SourceArn`–资源的亚马逊资源名称 (ARN)（请参阅“亚马逊资源名称 (ARN) (p. 59)”）
- `aws:SourceIp`–请求者的 IP 地址，用于 IP 地址条件（请参阅“IP 地址 (p. 59)”）
- `aws:UserAgent`–有关请求者客户端应用程序的信息，用于字符串条件（请参阅“字符串条件 (p. 57)”）
- `aws:userid`–检查请求者的用户 ID（请参阅“字符串条件 (p. 57)”）。
- `aws:username`–检查请求者的用户名称（请参阅“字符串条件 (p. 57)”）。

密钥名称不区分大小写。例如，`aws:CurrentTime` 等同于 `AWS:currenttime`。

Note

如果您使用 `aws:SourceIp` 并且请求源自 Amazon EC2 实例，那么我们会评估实例的公有 IP 地址以确定是否允许访问。

使用access policy language的每款 AWS 产品也提供特定于服务的密钥。有关可用的特定于服务的密钥列表，请参阅“Amazon SQS 策略的特别信息 (p. 64)”。

条件类型

这些是您可以规定的基本条件种类：

- 字符串
- 数值
- 日期和时间
- 布尔值
- IP 地址
- 亚马逊资源名称 (ARN)
- 条件密匙的存在

字符串条件

字符串条件让您使用字符串匹配规则进行限制。您使用的实际数据类型是字符串。

条件	描述
StringEquals	严格匹配 缩略版：streq
StringNotEquals	严格否定匹配 缩略版：strneq
StringEqualsIgnoreCase	严格匹配，忽略大小写 缩略版：streqi
StringNotEqualsIgnoreCase	严格否定匹配，忽略大小写 缩略版：strneqi
StringLike	区分大小写的松散匹配。这些值可以在字符串中的任何位置包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。 缩略版：strl
StringNotLike	不区分大小写的松散否定匹配。这些值可以在字符串中的任何位置包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。 缩略版：strnl

数字条件

数字条件可让您限制数字匹配规则的使用。整数或者十进数都可使用。不支持分数和无理数句法。

条件	描述
NumericEquals	严格匹配 缩略版：numeq
NumericNotEquals	严格否定匹配 缩略版：numneq
NumericLessThan	“小于”匹配 缩略版：numlt
NumericLessThanEquals	“小于或等于”匹配 缩略版：numlteq
NumericGreaterThan	“大于”匹配 缩略版：numgt
NumericGreaterThanEquals	“大于或等于”匹配 缩略版：numgteq

日期条件

日期条件让您通过日期和时间匹配规则进行限制。您必须通过 ISO 8601 日期格式中的一个 W3C 执行来指定全部日期/时间值（有关更多信息，请转到 http://www.w3.org/TR/NOTE-datetime ）。您可以将这些条件与 aws:CurrentTime 密钥搭配使用，以根据请求时间来限制访问。

Note

日期条件不允许使用通配符。

条件	描述
DateEquals	严格匹配 缩略版：dateeq
DateNotEquals	严格否定匹配 缩略版：dateneq
DateLessThan	密钥阻止生效的时间点 缩略版：datelt
DateLessThanEquals	密钥阻止生效的时间点 缩略版：datelteq
DateGreaterThan	密钥启动生效的时间点 缩略版：dategt
DateGreaterThanEquals	密钥启动生效的时间点 缩略版：dategteq

布尔值条件

条件	描述
Bool	严格布尔值匹配

IP 地址

IP 地址条件可以让您根据 IP 地址匹配规则进行限制。可以将其与 aws:SourceIp 密钥结合使用。该值必须采用标准的 CIDR 格式（例如 10.52.176.0/24）。有关更多信息，请转到 RFC 4632。

条件	描述
IpAddress	按照 IP 地址或范围许可
NotIpAddress	按照 IP 地址或范围拒绝

亚马逊资源名称 (ARN)

Amazon Resource Name 可以让您根据 ARN 匹配规则限制。您使用的实际数据类型是字符串。

条件	描述
ArnEquals	ARN 的严格匹配
ArnNotEquals	ARN 的严格否定匹配

条件	描述
ArnLike	不区分大小写的 ARN 松散匹配。ARN 的六个由冒号分隔开的部分都要单独检查，每一个部分都可包括一个多字符匹配通配符 (*) 或一个单字符匹配通配符。
ArnNotLike	不区分大小写的 ARN 松散否定匹配。这些值可以在字符串中的任何位置包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。

条件密钥的存在

使用 Null 条件检查授权时是否有条件密钥。在策略语句中使用 true（密钥不存在）或 false（密钥存在且值不为 null）。您可以使用这个条件来确定用户是否已经过 MFA 验证（多重验证）。例如，下列条件表明对于使用 Amazon EC2 API 的用户，MFA 身份验证必须存在（且*不为 null*）。

```
{ "Statement":[{ "Action":["ec2:*"], "Effect":"Allow", "Resource":["*"], "Con
dition":{ "Null":{"aws:MultiFactorAuthAge":"false"} } } ] }
```

受支持数据类型

本部分列出了access policy language支持的一组数据类型。语言不支持每个策略元素的所有类型（有关每个元素的支持的数据类型，请参阅“元素描述 (p. 52)”）。

access policy language支持以下数据类型：

- 字符串
- 数字（整数和浮动值）
- 布尔值
- 空
- 清单
- 映射
- 结构（仅为嵌套映射）

以下图表将各个数据类型映射至串行化。注意所有策略必须为 UTF-8。有关 JSON 数据类型的信息，请转到 RFC 4627。

类型	JSON
字符串	字符串
整数	数字
浮动值	数字
布尔值	真假
空	空
日期	字符串符合 W3C Profile of ISO 8601
IpAddress	字符串符合 RFC 4632
列表	数组

类型	JSON
数据元	数据元

Amazon SQS 策略示例

本部分显示了 Amazon SQS 常用案例的示例策略。

以下示例策略向 AWS 账户号为 111122223333 的开发人员授予对美国东部（弗吉尼亚北部）地区中名为 444455556666/queue1 的队列的 SendMessage 权限。

```
{ "Version": "2012-10-17", "Id": "Queue1_Policy_UUID", "Statement": {
"Sid":"Queue1_SendMessage", "Effect": "Allow", "Principal": { "AWS":
"111122223333" }, "Action": "sqs:SendMessage", "Resource": "arn:aws:sqs:us-east-
1:444455556666:queue1" } }
```

以下示例策略向 AWS 账号为 111122223333 的开发人员授予对名为 444455556666/queue1 的队列的 SendMessage 和 ReceiveMessage 权限。

```
{ "Version": "2012-10-17", "Id": "Queue1_Policy_UUID", "Statement": {
"Sid":"Queue1_Send_Receive", "Effect": "Allow", "Principal": { "AWS":
"111122223333" }, "Action": ["sqs:SendMessage","sqs:ReceiveMessage"], "Resource":
 "arn:aws:sqs:*:444455556666:queue1" } }
```

以下示例策略向两个不同的开发人员（AWS 账户号分别为 111122223333 和 444455556666）授予对美国东部（弗吉尼亚北部）地区中名为 123456789012/queue1 的队列使用 Amazon SQS 允许共享访问的所有操作的权限。

```
{ "Version": "2012-10-17", "Id": "Queue1_Policy_UUID", "Statement": {
"Sid":"Queue1_AllActions", "Effect": "Allow", "Principal": { "AWS":
["111122223333","444455556666"] }, "Action": "sqs:*", "Resource":
"arn:aws:sqs:us-east-1:123456789012:queue1" } }
```

以下示例策略向所有用户授予对名为 111122223333/queue1 的队列的 ReceiveMessage 权限。

```
{ "Version": "2012-10-17", "Id": "Queue1_Policy_UUID", "Statement": {
"Sid":"Queue1_AnonymousAccess_ReceiveMessage", "Effect": "Allow", "Principal":
 { "AWS": "*" }, "Action": "sqs:ReceiveMessage", "Resource":
"arn:aws:sqs:*:111122223333:queue1" } }
```

以下示例策略向所有用户授予对名为 111122223333/queue1 的队列的 ReceiveMessage 权限，但此权限仅在 2009 年 1 月 31 日中午至下午 3 点之间有效。

```
{ "Version": "2012-10-17", "Id": "Queue1_Policy_UUID", "Statement": {
"Sid":"Queue1_AnonymousAccess_ReceiveMessage_TimeLimit", "Effect": "Allow",
"Principal": { "AWS": "*" }, "Action": "sqs:ReceiveMessage", "Resource":
"arn:aws:sqs:*:111122223333:queue1", "Condition" : { "DateGreaterThan" : {
"aws:CurrentTime":"2009-01-31T12:00Z" }, "DateLessThan" : { "aws:Current
Time":"2009-01-31T15:00Z" } } } }
```

以下示例策略向所有用户授予对名为 111122223333/queue1 的队列使用可以共享的所有可能的 Amazon SQS 操作的权限，但条件是请求必须来自于 192.168.143.0/24 范围。

```
{ "Version": "2012-10-17", "Id": "Queue1_Policy_UUID", "Statement": {
"Sid":"Queue1_AnonymousAccess_AllActions_WhitelistIP", "Effect": "Allow",
"Principal": { "AWS": "*" }, "Action": "sqs:*", "Resource":
```

```
"arn:aws:sqs:*:111122223333:queue1", "Condition" : { "IpAddress" : {
"aws:SourceIp":"192.168.143.0/24" } } } }
```

以下策略示例具有两项陈述：

- 一项为 192.168.143.0/24 范围（192.168.143.188 除外）内的所有用户授予权限，允许上述用户对名为 111122223333/queue1 的队列使用 SendMessage 操作。
- 一组语句是阻止 10.1.2.0/24 范围内的所有用户使用该队列。

```
{ "Version": "2012-10-17", "Id": "Queue1_Policy_UUID", "Statement": [ {
"Sid":"Queue1_AnonymousAccess_SendMessage_IPLimit", "Effect": "Allow", "Princip
al": { "AWS": "*" }, "Action": "sqs:SendMessage", "Resource":
"arn:aws:sqs:*:111122223333:queue1", "Condition" : { "IpAddress" : {
"aws:SourceIp":"192.168.143.0/24" }, "NotIpAddress" : {
"aws:SourceIp":"192.168.143.188/32" } } }, { "Sid":"Queue1_AnonymousAccess_Al
lActions_IPLimit_Deny", "Effect": "Deny", "Principal": { "AWS": "*" }, "Action":
 "sqs:*", "Resource": "arn:aws:sqs:*:111122223333:queue1", "Condition" : {
"IpAddress" : { "aws:SourceIp":"10.1.2.0/24" } } } ] }
```

以下示例策略实现了亚马逊资源名称 (ARN) `arn:aws:sns:us-east-1:111122223333:test-topic` 指定的 Amazon Simple Notification Service 主题与名为 `arn:aws:sqs:us-east-1:111122223333:test-topic-queue` 的队列之间的连接。

```
{ "Version": "2012-10-17", "Id": "SNStoSQS", "Statement": { "Sid":"rule1",
"Effect": "Allow", "Principal": { "AWS": "*" }, "Action": "sqs:*", "Resource":
 "arn:aws:sqs:us-east-1:111122223333:test-topic-queue", "Condition" : {
"StringEquals" : { "aws:SourceArn":"arn:aws:sns:us-east-1:111122223333:test-
topic" } } } }
```

Amazon SQS 策略的特别信息

以下列表提供了特定于访问控制的 Amazon SQS 实施的信息。

- Amazon SQS 允许您仅共享某些类型的权限（有关更多信息，请参阅“了解权限 (p. 37)”）
- 每个策略都必须仅覆盖单一队列（在编写一个策略时，请勿包括覆盖不同队列的语句）
- 每个策略必须有一个唯一的策略 ID (`Id`)
- 策略中的每个语句必须有一个唯一的语句 ID (`sid`)
- 在您编写条件时，Amazon SQS 不会实施任何要使用的特殊密钥；可用的密钥只是常规 AWS 范围内的密钥。

以下表格列举了策略信息的最大限值。

姓名	最大限值
字节数	8192
语句	20
委托人	50
条件	10

使用 AWS Identity and Access Management (IAM) 进行访问控制

Topics

Amazon SQS 拥有自己的基于资源的权限系统，该系统使用了以用于 AWS Identity and Access Management (IAM) 策略的同一语言编写的策略。这意味着，使用 Amazon SQS 策略可以达到与使用 IAM 策略（例如，在 IAM 策略中使用变量）相同的效果。有关更多信息，请参阅《*使用 IAM*》指南中的“策略变量”部分。

使用 Amazon SQS 策略与使用 IAM 策略的主要区别在于：您可以使用 Amazon SQS 策略向另一个 AWS 账户授予访问您队列的权限，而使用 IAM 策略则不行。

Note

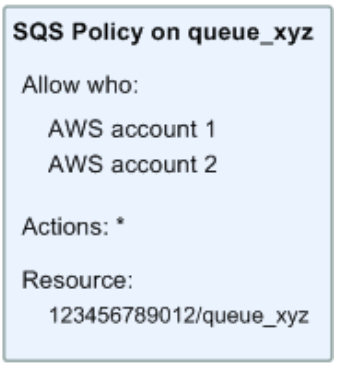
当您向其他 AWS 账户授予访问您 AWS 资源的权限时，请注意所有 AWS 账户都可以向其账户下的用户授予权限。这称为交叉账户访问。交叉账户访问能让你共享 AWS 资源，不需要管理新增的用户。有关使用交叉账户访问的信息，请参阅*使用 IAM*中的“启用交叉账户访问”部分。

本部分介绍了 Amazon SQS 策略系统如何与 IAM 搭配工作。

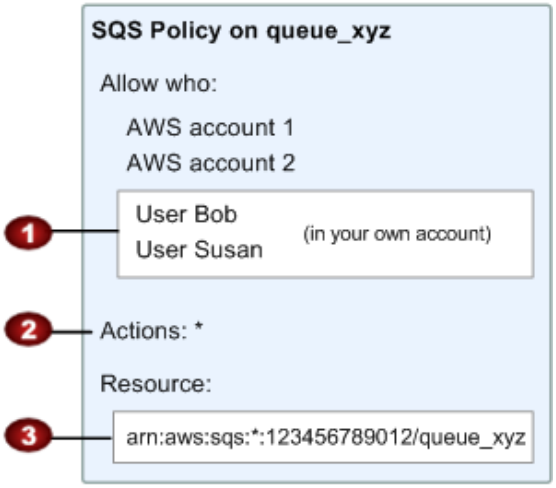
Amazon SQS 策略的 IAM 相关功能

您可以对队列使用 Amazon SQS 策略，以指定哪些 AWS 账户拥有访问该队列的权限。您可以指定访问类型和条件（例如，使用 `SendMessage` 和 `ReceiveMessage` 的权限；如果请求早于 2010 年 12 月 31 日）。您可以为其授予权限的特定操作是整个 Amazon SQS 操作列表的子集。如果您编写 Amazon SQS 策略并指定 * 以表示“所有 Amazon SQS 操作”，即表示该子集中的所有操作。

下图说明了这些基本 Amazon SQS 策略中涵盖操作子集的一个策略的概念。该策略用于 queue_xyz，并且向 AWS 账户 1 和 AWS 账户 2 授予对队列使用任何允许的操作的权限。请注意，该策略中的资源被指定为 123456789012/queue_xyz（其中，123456789012 是拥有该队列的账户的 AWS 账户 ID）。



随着 IAM 以及*用户* 和*亚马逊资源名称 (ARN)* 概念的推出，SQS 策略发生了一些变化。以下示意图和表格描述了这些变化。



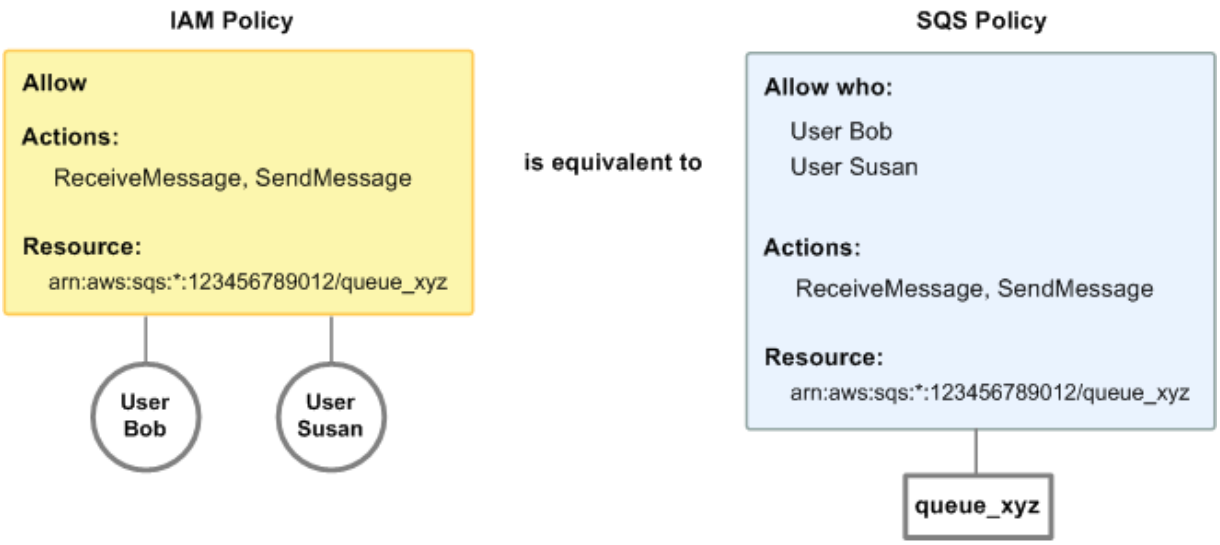
1	除了指定哪些 AWS 账户拥有访问该队列的权限以外，您还可以指定*您自己的 AWS 账户中* 哪些用户拥有访问该队列的权限。 这些用户不能位于另一个 AWS 账户中。
2	“*”中包含的操作子集已扩展（有关允许的操作的列表，请参阅“Amazon SQS 操作 (p. 69)”）。
3	您可以使用*亚马逊资源名称 (ARN)* 指定资源，这是您在 IAM 策略中必须指定资源的方法。有关 Amazon SQS 队列的 ARN 格式的信息，请参阅“Amazon SQS ARN (p. 69)”。 您仍可以改用原始格式 (<account_ID>/<queue_name>)。

例如，根据上图中所示的 Amazon SQS 策略，拥有 AWS 账户 1 或 AWS 账户 2 的安全证书的任何人都可以访问 queue_xyz。此外，您自己的 AWS 账户（ID 为 123456789012）中的用户 Bob 和 Susan 也可以访问该队列。

在推出 IAM 之前，Amazon SQS 会自动向某个队列的创建者授予对该队列的完全控制权限（例如，访问针对该队列的所有可能的 Amazon SQS 操作）。除非创建者使用的是 AWS 安全证书，否则上述情况将不会再出现。此外，任何有权创建队列的用户还必须有权使用其他 Amazon SQS 操作，这样才能对自己创建的队列执行任何操作。

IAM 和 Amazon SQS 策略一起使用

您可以使用以下两种方式向您的用户授予访问您的 Amazon SQS 资源的权限：通过 Amazon SQS 策略系统或 IAM 策略系统。您可以使用其中任意一套或两套系统。在绝大部分情况下，无论采用上述哪种方式，都可以得到同样的结果。例如，下图显示了等效的 IAM 策略和 Amazon SQS 策略。IAM 策略允许对您 AWS 账户中名为 queue_xyz 的队列执行 Amazon SQS `ReceiveMessage` 和 `SendMessage` 操作，并且它附加到用户 Bob 和 Susan（这意味着，Bob 和 Susan 拥有该策略中所述的权限）。此外，Amazon SQS 策略还向 Bob 和 Susan 授予对同一队列访问 `ReceiveMessage` 和 `SendMessage` 的权限。



Note

上述示例显示了不带任何条件的简单策略。您可以在上述任一策略中指定特定条件，并获得同样的结果。

IAM 和 Amazon SQS 策略之间有下面一点区别：Amazon SQS 策略系统允许您向其他 AWS 账户授予权限，而 IAM 则不允许。

将由您自己决定如何是否上述两种系统管理您的权限，您可以根据自身需求做出决定。以下示例展示这两种策略系统是如何共同运行的。

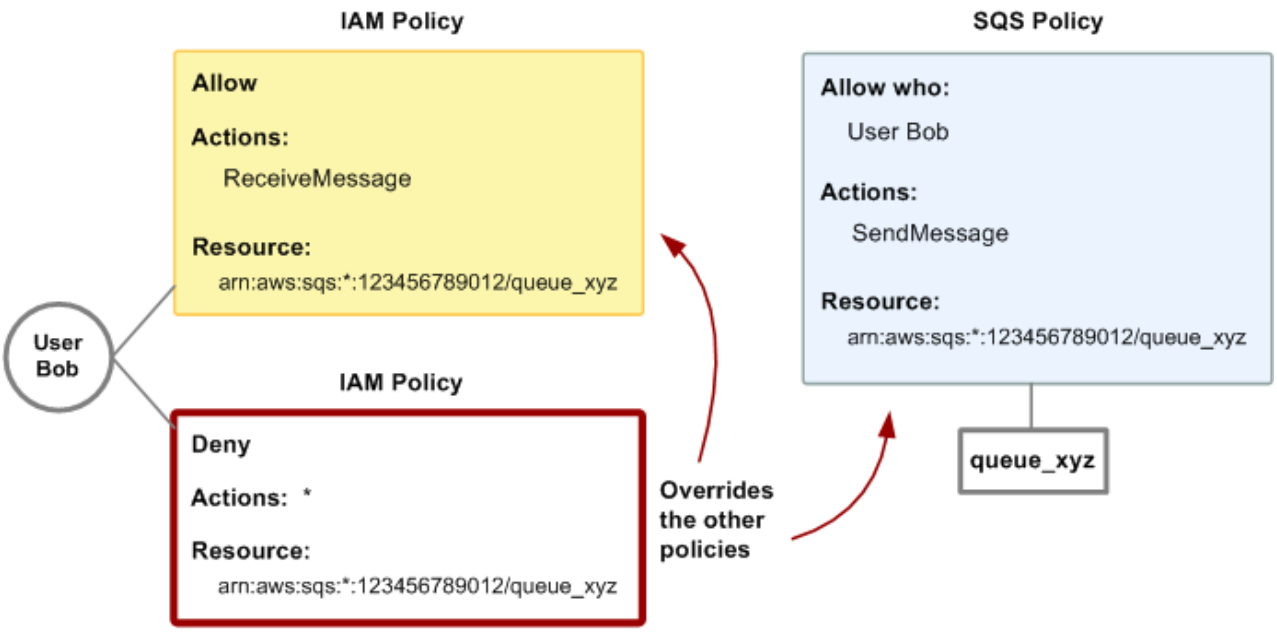
1

在本例中，Bob 同时拥有适用于他的 IAM 策略和 Amazon SQS 策略。IAM 策略向他授予对 queue_xyz 使用 `ReceiveMessage` 的权限，而 Amazon SQS 策略则向他授予对同一队列使用 `SendMessage` 的权限。下图阐明了这一概念。

如果 Bob 要发送从 queue_xyz 接收消息的请求，则 IAM 策略将允许该操作。如果 Bob 要发送向 queue_xyz 发送消息的请求，则 Amazon SQS 策略将允许该操作。

2

在本示例中，我们基于示例 1（其中，Bob 拥有两个适用于他的策略）来进行描述。假设 Bob 滥用他对 queue_xyz 的访问权限，因此，您希望删除他对该队列的所有访问权限。最简单的方法是添加一个拒绝他对该队列访问所有操作的策略。此第三个策略将覆盖其他两个策略，因为显式拒绝策略将始终覆盖允许策略（有关策略评估逻辑的更多信息，请参阅“评估逻辑 (p. 46)”）。下图阐明了这一概念。



或者，您可以向 Amazon SQS 策略中添加一条额外的语句，该语句拒绝 Bob 以任何方式访问该队列。此操作与添加拒绝他访问该队列的 IAM 策略具有同样的效果。

有关涉及 Amazon SQS 操作和资源的策略示例，请参阅“Amazon SQS 的 IAM 策略示例 (p. 70)”。有关编写 Amazon SQS 策略的更多信息，请转到《Amazon Simple Queue Service 开发者指南》。

Amazon SQS ARN

对于 Amazon SQS，队列是您可以在策略中指定的唯一资源类型。以下是队列的亚马逊资源名称 (ARN) 格式：

```
arn:aws:sqs:region:account_ID:queue_name
```

有关 ARN 的更多信息，请访问《*使用 IAM*》中的“IAM ARN”部分。

以下是 美国东部（弗吉尼亚北部）地区中名为 my_queue 的队列的 ARN，它属于 AWS 账户 123456789012。

```
arn:aws:sqs:us-east-1:123456789012:my_queue
```

如果您在 Amazon SQS 支持的各个不同地区中均有一个名为 my_queue 的队列，则您可以使用以下 ARN 指定这些队列。

```
arn:aws:sqs:*:123456789012:my_queue
```

您可以在队列名称中使用 * 和 ? 通配符。例如，以下语句可以引用 Bob 创建的所有队列，这些队列的前缀为 bob_。

```
arn:aws:sqs:*:123456789012:bob_*
```

为了方便起见，Amazon SQS 有一个名为 Arn 的队列属性，其值为队列的 ARN。您可以通过调用 Amazon SQS GetQueueAttributes 操作来获取该值。

Amazon SQS 操作

您在策略中指定的所有 Amazon SQS 操作都必须以小写字符串 Amazon SQS: 作为前缀。例如，Amazon SQS:CreateQueue。

在推出 IAM 之前，您可以对队列使用 Amazon SQS 策略，以指定哪些 AWS 账户拥有访问该队列的权限。此外，您还可以指定访问类型（例如，Amazon SQS:SendMessage、Amazon SQS:ReceiveMessage 等）。您可以为其授予权限的特定操作是整个 Amazon SQS 操作集的子集。如果您编写 Amazon SQS 策略并指定 * 以表示“所有 Amazon SQS 操作”，即表示该子集中的所有操作。该子集最初包括：

- Amazon SQS:SendMessage
- Amazon SQS:ReceiveMessage
- Amazon SQS:ChangeMessageVisibility
- Amazon SQS:DeleteMessage
- Amazon SQS:GetQueueAttributes（适用于除 Policy 以外的所有属性）

随着 IAM 的推出，该操作列表扩展到包括以下操作：

- Amazon SQS:CreateQueue
- Amazon SQS:DeleteQueue

- `Amazon SQS:ListQueues`

与向队列授予权限以及从队列删除权限相关的操作（`Amazon SQS:AddPermission` 等）已保留，因此没有显示在前面的两个列表中。这意味着，AWS 账户中的*用户*不能使用这些操作。但是，*AWS 账户*可以使用这些操作。

Amazon SQS 密钥

Amazon SQS 实施了以下策略密钥，但没有实施其他策略密钥。有关策略密钥的更多信息，请参阅"条件 (p. 55)"。

适用整个 AWS 范围的策略密钥

- `aws:CurrentTime`–检查日期/时间条件。
- `aws:EpochTime`–使用纪元日期或 UNIX 时间检查日期/时间条件。
- `aws:MultiFactorAuthAge`–使用 Multi-Factor Authentication (MFA) 查看多久之前下发了经 MFA 验证的发出请求之安全证书（以秒计）。与其他密钥不同，若 MFA 未使用，则表示该密钥不存在。
- `aws:principaltype`–检查当前请求的委托人类型（用户、账户、联合用户等）。
- `aws:SecureTransport`–检查请求是否是使用 SSL 发送的。对于仅使用 SSL 的服务（如 Amazon RDS 和 Amazon Route 53），`aws:SecureTransport` 密钥没有意义。
- `aws:SourceArn`–使用来源的亚马逊资源名称 (ARN) 查看请求的来源。（该值仅适用于某些产品。有关更多信息，请参阅 *Amazon Simple Queue Service 开发者指南* 中"Element Descriptions"部分的 Amazon Resource Name (ARN)。）
- `aws:SourceIp`–检查请求者的 IP 地址。请注意，如果您使用 `aws:SourceIp`，并且请求是来自 Amazon EC2 实例，则会评估实例的公有 IP 地址。
- `aws:UserAgent`–检查发出请求的客户端应用程序。
- `aws:userid`–检查请求者的用户 ID。
- `aws:username`–检查请求者的用户名称（如果可用）。

Note

密钥名称区分大小写。

Amazon SQS 的 IAM 策略示例

此部分演示了用于控制用户访问 Amazon SQS 的几个简单的 IAM 策略。

Note

未来，根据策略陈述的目标，Amazon SQS 可能会添加逻辑上包含在以下策略之一中的新操作。

1:允许用户创建和使用自己的队列

在本例中，我们为 Bob 创建了一个策略，该策略允许他访问所有 Amazon SQS 操作，但是仅限于针对名称以文本字符串 `bob_queue` 开头的队列。

Note

Amazon SQS 不会自动向队列创建者授予随后使用该队列的权限。因此，在我们的 IAM 策略中，除了 `CreateQueue` 以外，我们还必须向 Bob 显式授予使用所有 Amazon SQS 操作的权限。

```
{ "Version": "2012-10-17", "Statement":[{ "Effect":"Allow", "Action":"sqs:*",
"Resource":"arn:aws:sqs:*:123456789012:bob_queue*" } ] }
```

2:允许开发人员向共享测试队列写入消息

在本例中，我们为开发人员创建了一个组，并附加了一个策略，该策略允许该组使用 Amazon SQS `SendMessage` 操作，但是仅限于针对名为 *CompanyTestQueue* 的 AWS 账户队列。

```
{ "Version": "2012-10-17", "Statement":[{ "Effect":"Allow", "Action":"sqs:SendMes
sage", "Resource":"arn:aws:sqs:*:123456789012:CompanyTestQueue" } ] }
```

3:允许管理人员获取队列的一般大小

在本例中，我们为管理人员创建了一个组，并附加了一个策略，该策略允许该组对所有 AWS 账户队列使用 Amazon SQS `GetQueueAttributes` 操作。

```
{ "Version": "2012-10-17", "Statement":[{ "Effect":"Allow", "Ac
tion":"sqs:GetQueueAttributes", "Resource":"*" } ] }
```

4:允许合作伙伴向特定队列发送消息

您可以使用 Amazon SQS 策略或 IAM 策略来执行此操作。如果合作伙伴拥有 AWS 账户，则使用 Amazon SQS 策略可能更容易。但是，合作伙伴公司中拥有 AWS 安全证书的任何人（而不只是特定用户）都可以向该队列发送消息。假设您希望仅向特定人员（或应用程序）授予访问权限，则需要像对待您自己公司内的用户那样对待合作伙伴，并使用 IAM 策略（而不是 Amazon SQS 策略）。

在本示例中，我们创建了一个名为 WidgetCo 的组（代表合作伙伴公司），接着为需要访问权限的合作伙伴公司特定人员（或应用程序）创建了一个用户，然后将该用户放入该组。

随后，我们附加了一个策略，该策略向该组授予对名为 *WidgetPartnerQueue* 的特定队列的 `SendMessage` 访问权限。

此外，我们还希望阻止 WidgetCo 组对队列执行其他任何操作，因此，我们添加了一条语句，该语句拒绝该组对 WidgetPartnerQueue 以外的任何队列执行 `SendMessage` 以外的任何 Amazon SQS 操作。只有在系统中的其他地方存在广泛策略（该策略向用户授予广泛访问 Amazon SQS 的权限）时，才需要执行此操作。

```
{ "Version": "2012-10-17", "Statement":[{ "Effect":"Allow", "Action":"sqs:SendMes
sage", "Resource":"arn:aws:sqs:*:123456789012:WidgetPartnerQueue" }, { "Ef
fect":"Deny", "NotAction":"sqs:SendMessage", "NotRe
source":"arn:aws:sqs:*:123456789012:WidgetPartnerQueue" } ] }
```

使用临时安全证书

除了创建有其自己的安全证书的 IAM 用户之外，IAM 还可以让您向任何用户授予临时安全证书，从而使此用户可以访问您的 AWS 服务和资源。您可以管理有 AWS 账户的用户；这些用户是 IAM 用户。您还可以对您系统中没有 AWS 账户的用户进行管理；这些用户被称为联合用户。此外，“用户”还可以是您创建的能访问您的 AWS 资源的应用程序。

您可以使用上述临时安全证书对 Amazon SQS 发出请求。API 库将通过上述证书计算必要签名值以验证您的请求。如果您使用过期证书发送请求，Amazon SQS 会拒绝该请求。

首先，使用 IAM 创建临时安全证书，该证书包含安全令牌、访问密钥 ID 和私有访问密匙。其次，准备要使用临时访问密钥 ID 和安全令牌进行签名的字符串。然后，使用临时私有访问密钥（而不是您自己的私有访问密钥）对您的查询 API 请求签名。最后，在提交签名的查询 API 请求时，别忘了使用临时访问密钥 ID（而不是您自己的访问密钥 ID），并且包含安全令牌。有关 IAM 对临时安全证书的支持的更多信息，请转到《*使用 IAM*》中的"授予对您 AWS 资源的临时访问权限"部分。

使用临时安全证书调用 Amazon SQS 查询 API 操作

1. 使用 AWS Identity and Access Management 请求临时安全令牌。有关更多信息，请转到《使用 IAM》中的"创建临时安全证书，为 IAM 用户启用访问"部分。
   IAM 会返回一个安全令牌、一个访问密钥 ID 和一个私有访问密匙。
2. 像通常那样准备查询，但要使用临时访问密钥 ID（而不是您自己的访问密钥 ID），并且包含安全令牌。使用临时私有访问密钥（而不是您自己的私有访问密钥）对请求签名。
3. 提交已使用临时访问密钥 ID 和安全令牌签名的查询字符串。

   以下示例展示了如何使用临时安全证书对 Amazon SQS 请求进行身份验证。

```
http://sqs.us-east-1.amazonaws.com/ ?Action=CreateQueue &DefaultVisibility
Timeout=40 &QueueName=testQueue &Attribute.1.Name=VisibilityTimeout &Attrib
ute.1.Value=40 &Version=2011-10-01 &Signature=Dqlp3Sd6ljTUA9Uf6SGtEEx
wUQEXAMPLE &SignatureVersion=2 &SignatureMethod=HmacSHA256 &Expires=2011-
10-18T22%3A52%3A43PST &SecurityToken=SecurityTokenValue
&AWSAccessKeyId= Access Key ID provided by AWS Security Token Service
```

   以下示例使用了临时安全证书来通过 SendMessageBatch 发送两条消息。

```
http://sqs.us-east-1.amazonaws.com/ ?Action=SendMessageBatch &SendMessage
BatchRequestEntry.1.Id=test_msg_001 &SendMessageBatchRequestEntry.1.Message
Body=test%20message%20body%201 &SendMessageBatchRequestEntry.2.Id=test_msg_002
 &SendMessageBatchRequestEntry.2.MessageBody=test%20message%20body%202
&SendMessageBatchRequestEntry.2.DelaySeconds=60 &Version=2011-10-01 &Ex
pires=2011-10-18T22%3A52%3A43PST &Signature=Dqlp3Sd6ljTUA9Uf6SGtEExwUQEXAMPLE
 &SignatureVersion=2 &SignatureMethod=HmacSHA256 &SecurityToken=Security
TokenValue
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

使用 Amazon CloudWatch 监控 Amazon SQS

Amazon SQS 和 Amazon CloudWatch 相集成，因此，您可以使用 Amazon CloudWatch 来轻松收集、查看和分析 Amazon SQS 队列的指标。为 Amazon SQS 配置 Amazon CloudWatch 后，您可以更好地了解您的 Amazon SQS 队列和应用程序的性能。例如，您可以监控 NumberOfEmptyReceives 指标，以确保您的应用程序不会耗费太多时间来轮询新消息。此外，您还可以设置一个警报，使其在 Amazon SQS 指标（例如，NumberOfMessagesReceived）达到指定的阈值时，向您发送电子邮件通知。有关 Amazon SQS 发送到 Amazon CloudWatch 的所有指标的列表，请参阅“Amazon SQS 指标 (p. 74)”。

您使用 Amazon CloudWatch 为 Amazon SQS 队列配置的指标将每五分钟自动收集一次并推送到 Amazon CloudWatch。系统会对满足 Amazon CloudWatch 认为队列有效的准则的所有队列收集这些指标。从某个队列上次活动（即，任何 API 调用）以来最长六小时内，Amazon CloudWatch 会认为该队列有效。

Note

Amazon CloudWatch 中报告的 Amazon SQS 指标是不收费的；它们是作为 Amazon SQS 产品的一部分提供的。

访问 Amazon SQS 的 Amazon CloudWatch 指标

您可以使用 Amazon CloudWatch 控制台、Amazon CloudWatch 自己的命令行界面 (CLI) 或者以编程方式使用 Amazon CloudWatch API 来监控 Amazon SQS 的指标。下列程序告诉您使用不同的方式如何获得指标。

使用 Amazon CloudWatch 控制台查看指标

1. 登录 AWS 管理控制台，并通过以下网址打开 Amazon CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 单击 查看指标。
3. 从 正在查看 下拉菜单中选择 SQS:队列指标 以显示每个队列的可用指标。
4. 单击 MetricName 列中的特定指标以查看更多详细信息，例如所收集数据的图表。

从 Amazon CloudWatch CLI 访问指标

- 调用 `mon-get-stats`。您可以在《Amazon CloudWatch 开发者指南》中了解更多有关此函数以及其他与指标相关的函数的信息。

从 Amazon CloudWatch API 访问指标

- 调用 `GetMetricStatistics`。您可以在《Amazon CloudWatch API 参考》中了解更多有关此函数以及其他与指标相关的函数的信息。

为 Amazon SQS 指标设置 Amazon CloudWatch 警报

此外，Amazon CloudWatch 还允许您设置指标达到阈值时的警报。例如，您可能会为指标 NumberOfEmptyReceives 设置警报，这样，如果该指标在采样周期内达到您指定的阈值数字，则系统会发送电子邮件通知以告知您该事件。

使用 Amazon CloudWatch 控制台设置警报

1. 登录 AWS 管理控制台，并通过以下网址打开 Amazon CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 单击 警报,然后单击 创建警报 按钮。创建警报向导 将随即启动。
3. 滚动 Amazon SQS 指标以找到您要为其设置警报的指标。选择要为其创建警报的指标，然后单击继续。
4. 填写指标的 名称、描述、阈值 和 时间 值，然后单击 继续。
5. 按照警报状态选择 警报。如果您希望 Amazon CloudWatch 在达到警报状态时向您发送一封电子邮件，则您可以选择先前存在的 Amazon SNS 主题也可以单击 新建电子邮件主题。如果您单击新建电子邮件主题，则您可以为一个新的主题设置名称和电子邮件地址。此清单将会被保存下来并在将来报警器的下列框显示。单击继续。

   Note

   如果您使用新建电子邮件主题 创建新的 Amazon SNS 主题，那么电子邮件地址在接收通知之前必须通过验证。当报警器进入报警状态时，才发送电子邮件。如果在电子邮件地址验证之前报警状态发生变化，那么他们不会收到通知。

6. 此时，创建警报向导 会给您一次机会检查您即将创建的警报。如果您需要进行任何更改，则可以使用右侧的 编辑 链接。感觉满意后，单击 创建警报。

有关使用 Amazon CloudWatch 和警报的更多信息，请参阅“Amazon CloudWatch 文档”

Amazon SQS 指标

Amazon SQS 会向 Amazon CloudWatch 发送以下指标。

指标	描述
NumberOfMessagesSent	添加到队列的消息数量。 单位：*数量* 有效统计数据：总计
SentMessageSize	添加到队列的消息大小。 单位：*字节数* 有效统计数据：最小值、最大值、平均值和计数
NumberOfMessagesReceived	调用 ReceiveMessage API 操作返回的消息数量。 单位：*数量* 有效统计数据：总计
NumberOfEmptyReceives	未返回消息的 ReceiveMessage API 调用数量。 单位：*数量* 有效统计数据：总计
NumberOfMessagesDeleted	从队列删除的消息数量。 单位：*数量* 有效统计数据：总计
ApproximateNumberOfMessagesDelayed	队列中延迟且无法立即读取的消息数量。如果队列被配置为延迟队列，或者使用了延迟参数来发送消息，则会出现这种情况。 单位：*数量* 有效统计数据：平均值
ApproximateNumberOfMessagesVisible	可从队列取回的消息数量。 单位：*数量* 有效统计数据：平均值
ApproximateNumberOfMessagesNotVisible	“处于飞行状态”的消息数量。如果消息已发送到客户端，但尚未删除或尚未到达其可见性窗口末尾，则消息被认为处于飞行状态。 单位：*数量* 有效统计数据：平均值

附录 A：通过水平扩展和批处理提高吞吐量

作者：Marc Levy，2012 年 7 月

Amazon SQS 队列可以实现极高的吞吐量（每秒数千条消息）。达到此吞吐量的关键是水平扩展消息的创建者和使用者。此外，您还可以使用 Amazon SQS API 中的批处理操作来一次发送、接收或删除多达 10 条消息。在结合水平扩展后，批处理可以达到给定的吞吐量，而线程、连接和请求的数量却比单独的消息请求所需的数量更少。此外，由于 Amazon SQS 按请求（而不是消息）收费，因此，批处理还可以大幅降低费用。

本附录详细讨论了水平扩展和批处理，然后介绍了您自己可以尝试的简单示例。此外，本附录还简要讨论了您可以通过使用 Amazon CloudWatch 来监控的 Amazon SQS 吞吐量指标。

水平扩展

由于您通过 HTTP 请求-响应协议访问 Amazon SQS，因此，请求延迟（发起请求和接收响应之间的时间间隔）会限制您可以通过单一连接从单一线程达到的吞吐量。例如，如果从基于 Amazon Elastic Compute Cloud (Amazon EC2) 的客户端到同一地区内的 Amazon SQS 的延迟时间平均为大约 20ms，则通过单一连接从单一线程达到的最大吞吐量平均为每秒 50 次操作。

水平扩展意味着增加消息创建者（发出 SendMessage 请求）和使用者（发出 ReceiveMessage 和 DeleteMessage 请求）的数量，以提高整个队列的吞吐量。您可以通过增加客户端上的线程数量以及/或者添加客户端来进行水平扩展。添加更多客户端后，基本上应该能实现队列吞吐量的线性增长。例如，如果将客户端数量加倍，则您将获得两倍的吞吐量。

Important

进行水平扩展时，您需要确保您使用的 Amazon SQS 具有足够的连接或线程，以支持将要发送请求和接收响应的并发消息创建者和使用者的数量。例如，默认情况下，AWS SDK for Java 的 AmazonSQSClient 类的实例最多会维持 50 个与 Amazon SQS 的连接。要创建其他并发创建者和使用者，您需要调整该限制。例如，在AWS SDK for Java 中，您可以使用下面一行代码来调整 `AmazonSQSClient` 数据元上允许的最大创建者和使用者线程数量：

```
AmazonSQS sqsClient = new AmazonSQSClient(credentials, new ClientConfig
uration().withMaxConnections(producerCount + consumerCount));
```

此外，对于SDK for Java 异步客户端 AmazonSQSAsyncClient，您还需要确保有足够的线程可用。有关更多信息，请参阅您所使用的开发工具包库的文档。

批处理

Amazon SQS API 中的批处理操作（SendMessageBatch 和 DeleteMessageBatch）于 2011 年 10 月 (WSDL 2011-10-01) 推出，这些操作可一次处理最多十条消息，进一步优化吞吐量。由于 ReceiveMessage 一次可以处理十条消息，因此，上述 API 中没有 `ReceiveMessageBatch` 操作。

批处理的基本理念是：在与服务的每次往返操作中执行更多工作（例如，使用单一 `SendMessageBatch` 请求发送多条消息），以及在批处理请求的多条消息中分配批处理操作的延迟时间，而不是接受单一消息（例如，SendMessage 请求）的整个延迟时间。由于每次往返操作都会执行更多工作，因此，批处理请求可以更高效地使用线程和连接，从而提高吞吐量。Amazon SQS 按请求收费，因此，如果由更少的请求来处理等量的消息，则费用可以大幅降低。此外，更少的线程和连接可以降低客户端资源使用率，并且可以通过使用更小或更少的主机执行相同的工作来降低客户端费用。

不过，批处理确实会让应用程序变得有点复杂。例如，应用程序必须先积累消息，然后才能发送消息，并且有时候必须花费较长的时间来等待响应，但是批处理在以下情况下可能很有效：

- 您的应用程序正在短时间内生成大量消息，因此，延迟时间决不会很长。
- 消息使用者从队列中自行获取消息，这与典型的消息创建者相反，后者需要发送消息来响应它们无法控制的事件。

Important

即使批处理中单独的消息失败了，批处理请求（`SendMessageBatch` 或 `DeleteMessageBatch`）也可能会成功。在提出批处理请求后，您应始终检查各条消息是否失败了，并在必要时重试。

示例

本节所示的示例会实施简单的创建者-使用者模式。可供免费下载的完整示例位于：https://s3.amazonaws.com/cloudformation-examples/sqs-producer-consumer-sample.tar。本节后面的部分描述了各个模板部署的资源。

/tmp/sqs-producer-consumer-sample/src 中的配置实例提供了示例代码。配置运行的命令行位于 /tmp/sqs-producer-consumer-sample/command.log 中。

主线程会生成大量创建者和使用者线程，这些线程会在指定时间内处理 1KB 消息。示例包括提出单一操作请求的创建者和使用者，以及提出批处理请求的其他创建者和使用者。

在程序中，每个创建者线程都会发送消息，直到主线程停止创建者线程为止。`producedCount` 数据元会跟踪所有创建者线程生成的消息数量。处理错误的操作很简单：如果存在错误，则程序会退出 `run()` 方法。默认情况下，对于因暂时性错误而失败的请求，AmazonSQSClient 会重试三次，因此，此类错误很少会出现。必要时，可以配置重试计数，以减少系统引发的异常数量。消息创建者上的 `run()` 方法实施如下：

```
try { while (!stop.get()) { sqsClient.sendMessage(new SendMessageRe
quest(queueUrl, theMessage)); producedCount.incrementAndGet(); } } catch
(AmazonClientException e) { // By default AmazonSQSClient retries calls 3 times
 before failing, // so when this rare condition occurs, simply stop. log.er
ror("Producer: " + e.getMessage()); System.exit(1); }
```

批处理创建者几乎相同。一个显著的区别是需要重试失败的各个批处理条目：

```
SendMessageBatchResult batchResult = sqsClient.sendMessageBatch(batchRequest);
 if (!batchResult.getFailed().isEmpty()) { log.warn("Producer: retrying sending
 " + batchResult.getFailed().size() + " messages"); for (int i = 0, n = bat
chResult.getFailed().size(); i < n; i++) sqsClient.sendMessage(new SendMessage
Request(queueUrl, theMessage)); }
```

使用者的 run() 方法如下：

```
while (!stop.get()) { result = sqsClient.receiveMessage(new ReceiveMessageRe
quest(queueUrl)); if (!result.getMessages().isEmpty()) { m = result.getMes
sages().get(0); sqsClient.deleteMessage(new DeleteMessageRequest(queueUrl,
m.getReceiptHandle())); consumedCount.incrementAndGet(); } }
```

每个使用者线程都会接收和删除消息，直到主线程停止使用者线程为止。consumedCount 数据元会跟踪所有使用者线程使用的消息数量，并且计数会定期记录。批处理使用者与此相似，不同之处是一次最多接收十条消息，并且它使用的是 DeleteMessageBatch，而不是 DeleteMessage。

运行示例

您可以使用提供的 AWS CloudFormation 模板以下列三种不同配置运行示例代码：单一主机和单一操作请求、两台主机和单一操作请求、一台主机和批处理请求。

Important

完整示例位于单一 .tar 文件中。本节后面的部分描述了各个模板部署的资源。

/tmp/sqs-producer-consumer-sample/src 中的配置实例提供了示例代码。配置运行的命令行位于 /tmp/sqs-producer-consumer-sample/command.log 中。

示例设置了默认持续时间（20 分钟），以提供容量指标的三个或四个 5 分钟 Amazon CloudWatch 数据点。每次运行的 Amazon EC2 费用将是 m1.large 实例费用。Amazon SQS 费用基于每个示例的 API 调用率而发生变化，并且该调用率应介于大约 38 000 次 API 调用/分（针对批处理示例）和 380 000 次 API 调用/分（针对 2 台主机的单一 API 示例）之间。例如，在单一主机上运行一次单一 API 示例应花费大约 1 实例小时的 m1.large（大型标准按需实例，自 2012 年 7 月起为 0.32 USD），而对于持续时间为默认的 20 分钟的 Amazon SQS 操作，则应花费大约 20 分钟 x 190 000 次 API 调用/分 x 1 USD/1 000 000 次 API 调用 = 3.80 USD（自 2012 年 7 月起的定价，请检查当前定价）。

如果您要在除美国东部（弗吉尼亚北部）地区以外的地区部署 AWS CloudFormation 堆栈，请在 AWS CloudFormation 控制台的“地区”框中单击所需的地区。

运行示例

1. 单击下面与您要启动的堆栈相对应的链接：

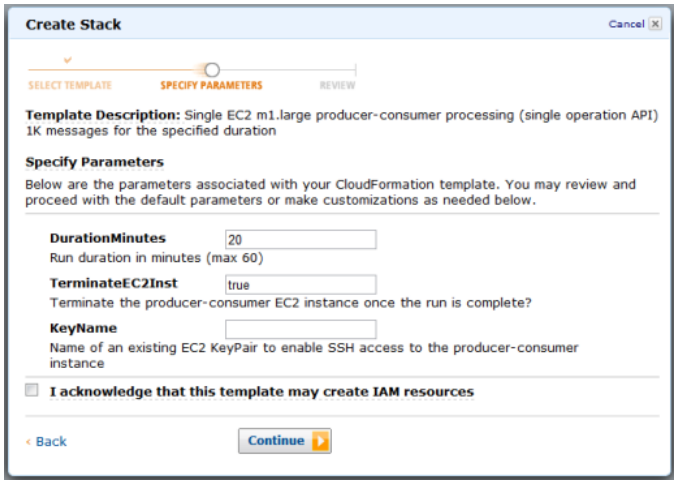
- 单一操作 API，一台主机：SQS_Sample_Base_Producer_Consumer.template 示例模板使用 Amazon SQS API 请求的单一操作形式：`SendMessage`、`ReceiveMessage` 和 `DeleteMessage`。单一 m1.large Amazon EC2 实例会启动 16 个创建者线程和 32 个使用者线程。

  要查看该模板，请转到 https://s3.amazonaws.com/cloudformation-templates-us-east-1/SQS_Sample_Base_Producer_Consumer.template
- 单一操作 API，两台主机：SQS_Sample_Base_Producer_Consumer_x2.template 示例模板使用 Amazon SQS API 请求的单一操作形式，但它使用的是两个（而不是一个）m1.large Amazon EC2 实例，每个实例都有 16 个创建者线程和 32 个使用者线程，于是总共有 32 个创建者和 64 个使用者。这说明了 Amazon SQS 在吞吐量与更大的创建者和使用者数量按比例提高时的弹性。

  要查看该模板，请转到 https://s3.amazonaws.com/cloudformation-templates-us-east-1/SQS_Sample_Base_Producer_Consumer_x2.template
- 批处理 API，一台主机：SQS_Sample_Batch_Producer_Consumer.template 示例模板在具有 12 个创建者线程和 20 个使用者线程的单一 m1.large Amazon EC2 实例上使用 Amazon SQS API 请求的批处理形式。

  要查看该模板，请转到 https://s3.amazonaws.com/cloudformation-templates-us-east-1/SQS_Sample_Batch_Producer_Consumer.template

2. 如果系统提示您，请登录到 AWS Management Console。
3. 在 创建堆栈 向导的 选择模板 页面上，单击 继续。
4. 在 指定参数 页面上，指定程序应运行的时长以及您是否希望在运行完成后自动终止 Amazon EC2 实例，并且提供 Amazon EC2 密钥对，以便您可以访问运行该示例的实例。示例如下：



5. 选中 我确认，此模板可创建 IAM 资源 复选框。所有模板都会创建 AWS Identity and Access Management (IAM) 用户，以便创建者-使用者程序可以访问队列。
6. 根据需要设定好所有设置后，单击 继续。
7. 在 审核 页面上，检查设置。如果这些设置与您期望的设置相符，请单击 继续。否则，请单击 返回 并进行必要的更改。
8. 在此向导的最后一个页面上，单击 关闭。堆栈部署可能需要花费几分钟时间。

要了解堆栈部署的进度，请在 AWS CloudFormation 控制台中单击示例堆栈。在下窗格中，单击 事件 选项卡。创建堆栈后，不到 5 分钟时间，示例就会开始运行。示例开始运行后，您可以在 Amazon SQS 控制台中查看队列。

要监控队列活动，您可以执行以下操作：

- 访问客户端实例，然后打开其针对迄今为止生成和使用的消息计数的输出日志文件 (/tmp/sqs-producer-consumer-sample/output.log)。此计数每秒更新一次。
- 在 Amazon SQS 控制台中，观察 可用的信息 和 传输中的消息 数字的变化。

此外，在启动队列后最多延迟 15 分钟，您就可以在 Amazon CloudWatch 中监控队列，如本主题后面部分中所述。

虽然模板和示例有安全措施来防止过度使用资源，但是在运行完示例后，您最好删除您的 AWS CloudFormation 堆栈。要执行此操作，请在 Amazon SQS 控制台中单击要删除的堆栈，然后单击 删除堆栈。删除所有资源后，Amazon CloudWatch 指标将全部下降为零。

监控示例运行的容量指标

Amazon SQS 会针对发送、接收和删除的消息自动生成容量指标。您可以通过 Amazon CloudWatch 控制台访问这些指标和其他指标。队列启动后，这些指标最多可能需要花费 15 分钟才可用。要管理搜索结果集，请单击 搜索，然后选中与您要监控的队列和指标相对应的复选框。

以下是针对这三个示例连续运行的 NumberOfMessageSent 指标。您的结果可能会有所不同，但这些结果在本质上应该是相似的：

- `NumberOfMessagesReceived` 和 `NumberOfMessagesDeleted` 指标显示了相同的模式，但我们在此图中省略了这两个指标，以减少凌乱。
- 第一个示例（单一 m1.large 上的单一操作 API）在 5 分钟内传递了大约 210 000 条消息（每秒传递了大约 700 条消息），并且接收和删除操作的吞吐量相同。
- 第二个示例（两个 m1.large 实例上的单一操作 API）传递了大约相当于该吞吐量两倍的吞吐量：在 5 分钟内传递了大约 440 000 条消息（每秒传递了大约 1 450 条消息），并且接收和删除操作的吞吐量相同。
- 最后一个示例（单一 m1.large 上的批处理 API）在 5 分钟内传递了 800 000 多条消息（每秒传递了大约 2 500 条消息），并且接收和删除的消息的吞吐量相同。如果批处理大小为 10，则系统会使用少得多的请求（并因此而以更低的费用）来处理这些消息。

附录 B：客户端缓冲和请求批处理

AWS SDK for Java (http://aws.amazon.com/sdkforjava/) 现在包含一个缓冲的异步客户端 AmazonSQSBufferedAsyncClient，用于访问 Amazon SQS。通过启用客户端缓冲（其中，从客户端发出的调用先进行缓冲，然后再作为批处理请求发送到 Amazon SQS），此新客户端允许您更轻松地对请求进行批处理。

客户端缓冲最多允许缓冲 10 个请求并将这些请求作为批处理请求发送，而不是分别发送各个请求。因此，随着发送到 Amazon SQS 的请求数量的减少，使用该产品的费用会降低。AmazonSQSBufferedAsyncClient 会缓冲同步和异步调用。此外，成批请求以及对长轮询的支持还可以帮助提高吞吐量（每秒传输的消息数量）。有关更多信息，请参阅“Amazon SQS 长轮询 (p. 10)”和“附录 A：通过水平扩展和批处理提高吞吐量 (p. 76)”。

从异步客户端 AmazonSQSAsyncClient 迁移到缓冲的异步客户端 AmazonSQSBufferedAsyncClient 只需要对现有的代码进行最少的更改。这是因为 AmazonSQSBufferedAsyncClient 会实施与 AmazonSQSAsyncClient 相同的接口。

AmazonSQSBufferedAsyncClient 入门

在开始使用本部分中的示例代码前，您必须先下载AWS SDK for Java (http://aws.amazon.com/sdkforjava/)。此外，请按照《AWS SDK for Java 入门》(http://aws.amazon.com/articles/3586) 中的步骤操作。

以下代码示例显示了如何基于 AmazonSQSAsyncClient 创建新的 AmazonSQSBufferedAsyncClient。

```
// Create an object representing your AWS credentials AWSCredentials credentials
 = new BasicAWSCredentials(myAccessKeyId, mySecretKey); // Create the basic
Amazon SQS async client AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient(cre
dentials); // Create the buffered client AmazonSQSAsync bufferedSqs = new
AmazonSQSBufferedAsyncClient(sqsAsync);
```

创建了新的 AmazonSQSBufferedAsyncClient 后，您可以像调用 AmazonSQSAsyncClient 那样调用该客户端，如以下代码示例所示。

```
CreateQueueRequest createRequest = new CreateQueueRe
quest().withQueueName("MyTestQueue"); CreateQueueResult res = bufferedSqs.cre
ateQueue(createRequest); SendMessageRequest request = new SendMessageRequest();
 String body = "test message_" + System.currentTimeMillis(); request.setMessage
Body( body ); request.setQueueUrl(res.getQueueUrl()); SendMessageResult
```

```
sendResult = bufferedSqs.sendMessage(request); ReceiveMessageRequest receiveRq
 = new ReceiveMessageRequest() .withMaxNumberOfMessages(1)
.withQueueUrl(queueUrl); ReceiveMessageResult rx = bufferedSqs.receiveMessage(re
ceiveRq);
```

高级配置

AmazonSQSBufferedAsyncClient 预配置了适用于大多数使用案例的设置。如果您想自己配置它，则可以使用 QueueBufferConfig 类进行配置。只需使用所需的设置创建一个 QueueBufferConfig 实例，并将该实例提供给 AmazonSQSBufferedAsyncClient 构造函数，如以下示例代码所示。

```
// Create an object representing your AWS credentials AWSCredentials credentials
 = new BasicAWSCredentials(myAccessKeyId, mySecretKey); // Create the basic
Amazon SQS async client AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient(cre
dentials); QueueBufferConfig config = new QueueBufferConfig() .withMaxInflightRe
ceiveBatches(5) .withMaxDoneReceiveBatches(15); // Create the buffered client
AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync, config);
```

可用于配置QueueBufferConfig 的参数如下：

- *longPoll*—如果此参数设置为 true，则 AmazonBufferedAsyncClient 会在检索消息时尝试使用长轮询。默认值为 true。

- *longPollWaitTimeoutSeconds*—在返回空接收结果前，接收消息调用在服务器上阻塞以等待消息显示在队列中的最长时间（以秒为单位）。如果禁用长轮询，则此设置不起作用。此设置的默认值为 20 秒。

- *maxBatchOpenMs*—传出调用等待其他同类调用进行批处理的最长时间（以毫秒为单位）。该设置越长，则执行等量工作所需的批处理就越少。当然，该设置越长，则批处理中的首次调用必须等待的时间就越长。如果此参数设置为零，则提交的请求不会等待其他请求，从而有效地禁用了批处理。此设置的默认值为 200 毫秒。

- *maxBatchSize*—将在单一批处理请求中一起进行批处理的最大消息数量。该设置越大，则执行等量请求所需的批处理就越少。此设置的默认值为每个批处理 10 个请求，这也是 Amazon SQS 当前允许的最大批处理大小。

- *maxBatchSizeBytes*—客户端尝试向 Amazon SQS 发送的消息批处理的最大大小（以字节为单位）。默认值为 64KB，这也是 Amazon SQS 当前允许的最大消息和批处理大小。

- *maxDoneReceiveBatches*—AmazonBufferedAsyncClient 预取并存储在客户端上的最大接收批处理数量。该设置越大，则不必调用 Amazon SQS 服务器即可满足的接收请求就越多。但是，预取的消息越多，则消息在缓冲区中停留的时间就越长，这意味着消息的可见性超时将会过期。如果此参数设置为零，则消息的所有预取操作都会禁用，并且系统仅按需取回消息。默认值为 10 个批处理。

- *maxInflightOutboundBatches*—可以同时处理的最大活跃出站批处理数量。该设置越大，则可以发送出站批处理的速度就越快（受限于其他限制，例如 CPU 或带宽）。该设置越大，则 AmazonSQSBufferedAsyncClient 占用的线程就越多。默认值为 5 个批处理。

- *maxInflightReceiveBatches*—可以同时处理的最大活跃接收批处理数量。该设置越大，则可以接收的消息就越多（受限于其他限制，例如 CPU 或带宽、命中）。尽管如此，但是该设置越大，则 AmazonSQSBufferedAsyncClient 占用的线程就越多。如果此参数设置为 0，则消息的所有预取操作都会禁用，并且系统仅按需取回消息。默认值为 10 个批处理。

- *visibilityTimeoutSeconds*—如果此参数设置为非零正值，则此可见性超时会覆盖从中检索消息的队列上设置的可见性超时。零秒的可见性超时不受支持。默认值为 -1，这意味着使用默认队列设置。

附录 C：为队列订阅 Amazon SNS 主题

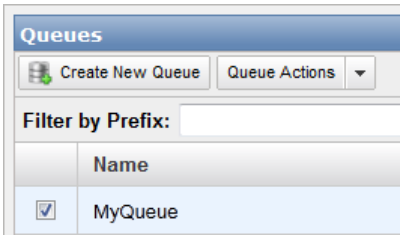
现在，您可以使用 Amazon SQS 的 AWS Management Console来为 Amazon SQS 队列订阅 Amazon SNS 主题，该控制台简化了此过程。例如，您可以从可用于选定队列的主题列表中进行选择。随后，Amazon SQS 会负责为队列订阅主题，并添加必要的权限。消息发布到主题后，Amazon SNS 会向订阅的队列发送 Amazon SQS 消息。有关 Amazon SNS 的更多信息，请参阅《Amazon SNS 入门》。有关 Amazon SQS 的更多信息，请参阅《Amazon SQS 入门》。

使用 AWS Management Console为队列订阅 Amazon SNS 主题

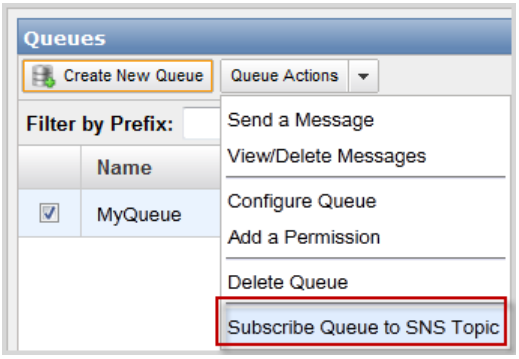
以下步骤假设您已经创建了队列和 Amazon SNS 主题。

使用 AWS Management Console为队列订阅 Amazon SNS 主题

1. 登录 AWS 管理控制台，并通过以下网址打开 Amazon SQS 控制台：https://console.aws.amazon.com/sqs/。
2. 选择要为其订阅 Amazon SNS 主题的队列。



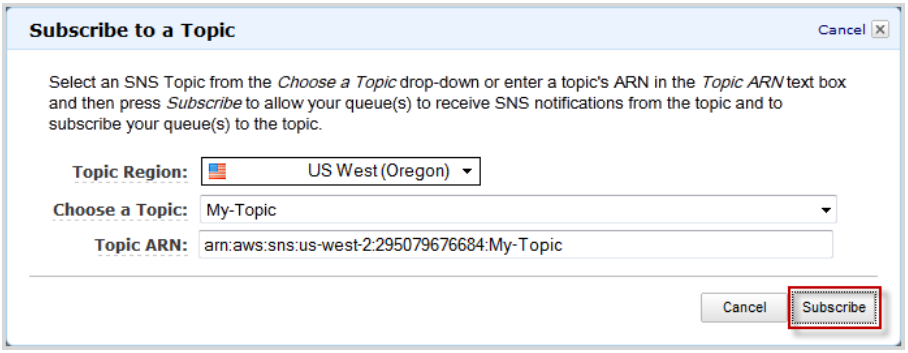
3. 从 队列操作 下拉列表中选择 为队列订阅 SNS 主题。



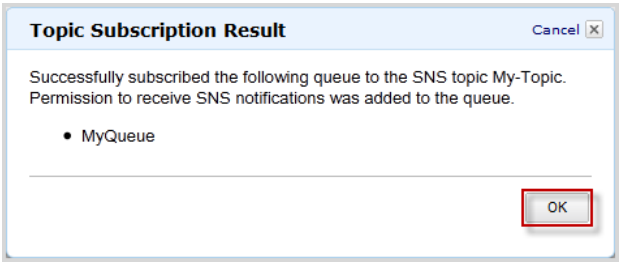
4. 从 选择一个主题 下拉列表中选择要为队列订阅的 Amazon SNS 主题，然后单击 订阅。

   Note

   此外，您还可以在 主题 ARN 框中输入 Amazon SNS 主题的 ARN。如果您希望从某个 AWS 账户（该账户不同于您用来创建该队列的账户）为队列订阅 Amazon SNS 主题，则此操作非常有用。此外，如果 Amazon SNS 主题未在 选择一个主题 下拉列表中列出，则此操作也非常有用。



5. 在 主题订阅结果 对话框中，单击 确定。



通过向主题发布消息，然后查看主题向队列发送的消息，您可以验证主题的队列订阅结果。有关详细步骤，请参阅“通过向主题发布消息，然后阅读来自队列的消息，对其进行测试”。

Amazon SQS 资源

下表列出了在您使用此服务时可为您提供帮助的相关资源。

资源	描述
Amazon Simple Queue Service Getting Started Guide	该入门指南基于简单的使用案例，提供了该服务的快速教程。其中包括使用多种编程语言编写的示例和说明。
Amazon Simple Queue Service API 参考	API 参考提供了 WSDL 位置；API 操作、参数和数据类型的完整描述；该服务返回的错误列表。
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.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon SQS.
AWS Premium Support Information	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.

文档历史记录

下表说明了自 *Amazon Simple Queue Service 开发人员指南*上次发布以来对该文档所做的重要更改。

- API 版本：2012-11-05
- 最近文档更新时间：2012 年 11 月 21 日

变更	描述	修改日期
新控制台功能	现在，您可以使用 Amazon SQS 的 AWS Management Console来为 Amazon SQS 队列订阅 Amazon SNS 主题，该控制台简化了此过程。有关更多信息，请参阅 附录 C：为队列订阅 Amazon SNS 主题 (p. 84)。	2012 年 11 月 21 日
新功能	Amazon SQS 的 2012-11-05 API 版本添加了对签名版本 4（该版本提高了安全性和性能）的支持。有关签名版本 4 的更多信息，请参阅查询请求身份验证 (p. 34)。	2012 年 11 月 5 日
新功能	AWS SDK for Java 现在包含一个缓冲的异步客户端 AmazonSQSBufferedAsyncClient，以便访问 Amazon SQS。通过启用客户端缓冲（其中，从客户端发出的调用先进行缓冲，然后再作为批处理请求发送到 Amazon SQS），此新客户端允许您更轻松地对请求进行批处理。有关客户端缓冲和请求批处理的更多信息，请参阅附录 B：客户端缓冲和请求批处理 (p. 81)。	2012 年 11 月 5 日
新功能	Amazon SQS 的 2012-11-05 API 版本添加了长轮询支持。长轮询允许 Amazon SQS 等待指定的时间，直到消息可用，而不是在消息不可用时返回空响应。有关长轮询的更多信息，请参阅“Amazon SQS 长轮询 (p. 10)”。	2012 年 11 月 5 日