aws

开发人员指南

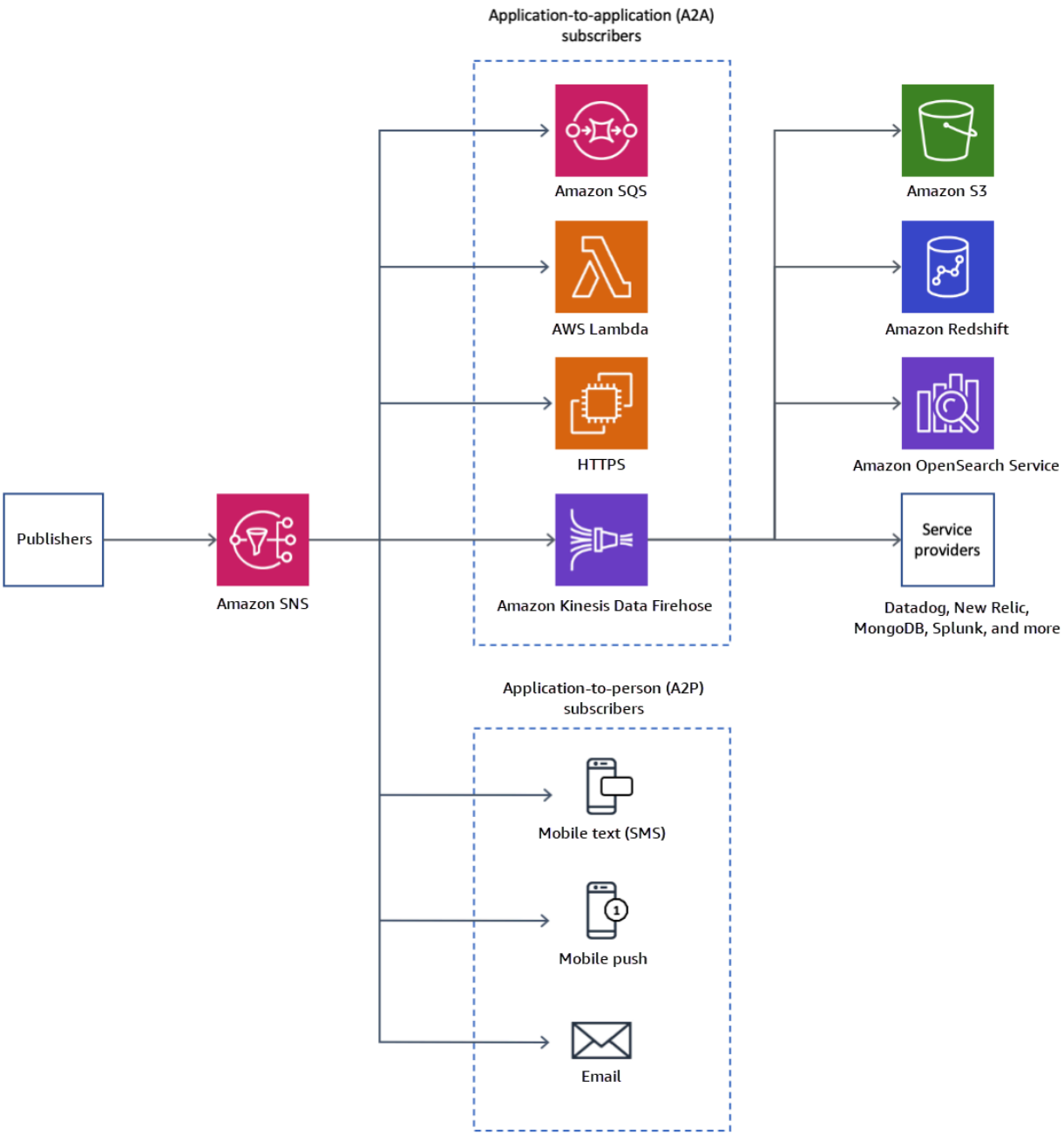
Amazon Simple Notification Service

Amazon Simple Notification Service: 开发人员指南

Table of Contents

什么是亚马逊SNS？

亚马逊简单通知服务 (AmazonSNS) 是一项托管服务，可从发布者向订阅者（也称为制作者和消费者）提供消息传送。发布者通过将消息发送至主题与订阅者进行异步交流，主题是一个逻辑访问点和通信渠道。客户可以使用支持的终端节点类型订阅亚马逊SNS主题并接收已发布的消息，例如 Amazon Data Firehose SQS、、 AWS Lambda、HTTP、、电子邮件、移动推送通知和移动短信 () SMS。



主题

- Amazon SNS 的特性和功能
- AWS Amazon 常用的服务 SNS
- 访问亚马逊 SNS

- 亚马逊的定价 SNS
- 常见的亚马逊SNS场景
- 将 Amazon SNS 与 AWS SDK

Amazon SNS 的特性和功能

Amazon SNS 提供以下特性和功能：

- 一条pplication-to-application 消息

  pplication-to-application 消息支持订阅者，例如 Amazon Data Firehose 交付流、Lambda 函数、亚马逊SQS队列、HTTP /S 终端节点和 AWS 事件分叉管道。有关更多信息，请参阅 pplication-to-application (A2A) 消息传递。

- A pplication-to-person 通知

  pplication-to-person 通知向订阅者提供用户通知，例如移动应用程序、移动电话号码和电子邮件地址。有关更多信息，请参阅 pplication-to-person 消息传递 (A2P)。

- 标准和FIFO主题

  使用FIFO主题来确保严格的消息顺序、定义消息组并防止消息重复。您可以同时使用标准队列FIFO和标准队列来订阅FIFO主题。有关更多信息，请参阅 使用FIFO主题对消息进行排序和重复数据删除。

  如果邮件传输顺序和可能的邮件重复并不重要，请使用标准主题。所有受支持的传输协议都可以订阅标准主题。

- 消息持久性

  Amazon SNS 使用多种策略协同工作来提供消息的持久性：

  - 已发布的消息存储在多个地理位置分隔的服务器和数据中心之间。
  - 如果订阅的终端节点不可用，Amazon SNS 将执行传输重试政策。
  - 要保留在传输重试策略结束之前未传输的任何消息，您可以创建死信队列。

- 消息归档、重播和分析

  您可以通过多种方式SNS在亚马逊存档消息，包括将 Firehose 传送流SNS订阅主题，这样您就可以向分析端点发送通知，例如亚马逊简单存储服务 (Amazon S3) 存储桶、Amazon Redshift 表等。此外，Amazon SNS FIFO 主题支持消息存档和重播为无代码、就地消息存档，允许主题所有者在其主

题中存储（或存档）消息。然后，主题订阅用户可以将归档的消息检索（或重播）回订阅的端点。有关更多信息，请参阅Amazon SNS 消息存档和重播FIFO主题。

- 消息属性

  消息属性让您可以提供有关消息的任意元数据。the section called “消息属性”。

- 消息筛选

  默认情况下，每个订阅者会收到发布到该主题的每条消息。要仅接收一部分消息，订阅者必须将筛选策略分配给主题订阅。订阅者还可以定义筛选策略范围，以启用基于有效负载或基于属性的筛选。筛选策略范围的默认值为 `MessageAttributes`。当传入消息属性与筛选策略属性匹配时，消息将传输到订阅的终端节点。否则，消息将被筛选掉。当筛选策略范围为 `MessageBody` 时，筛选策略属性将与有效负载进行匹配。有关更多信息，请参阅 消息筛选。

- 消息安全性

  服务器端加密使用提供的加密密钥保护存储在 Amazon SNS 主题中的消息内容。 AWS KMS有关更多信息，请参阅 the section called “使用服务器端加密保护数据”。

  您还可以在 Amazon SNS 和您的虚拟私有云 (VPC) 之间建立私有连接。有关更多信息，请参阅the section called “使用VPC端点保护流量”。

AWS Amazon 常用的服务 SNS

您可以在 Amazon 上使用以下服务SNS：

- Amazon SQS 提供安全、耐用且可用的托管队列，可让您集成和分离分布式软件系统和组件。亚马逊SQS通过以下方式与亚马逊SNS相关：
  - Amazon SNS 为无法送达的消息提供由 Amazon 支持的死信队列SQS。
  - 您可以通过亚马逊SQS队列订阅亚马逊SNS主题。
  - 您可以为亚马逊SQSFIFO队列或标准队列订阅亚马逊SNSFIFO主题。只有 Amazon SQS FIFO 队列才能保证消息按顺序接收，并且没有重复内容。
- AWS Lambda 可用于构建快速响应新信息的应用程序。在高可用性计算基础设施上运行 Lambda 函数中的应用程序代码。有关更多信息，请参见AWS Lambda 开发人员指南。您可以将 Lambda 函数订阅到某个主题。SNS
- AWS Identity and Access Management (IAM) 可帮助您安全地控制用户对 AWS 资源的访问权限。IAM用于控制谁可以使用您的 Amazon SNS 主题（身份验证）、他们可以使用哪些主题以及如何使用这些主题（授权）。有关更多信息，请参阅 在 Amazon 上使用基于身份的政策 SNS。

- AWS CloudFormation使您能够对 AWS 资源进行建模和设置。创建描述 AWS 所需资源的模板，包括 Amazon SNS 主题和订阅。 AWS CloudFormation 负责为您配置和配置这些资源。有关更多信息，请参阅 用户指南。AWS CloudFormation

访问亚马逊 SNS

您可以使用 Amazon SNS 控制台、命令行工具或配置和管理SNS主题和订阅 AWS SDKs。

- Amazon SNS 控制台提供了便捷的用户界面，用于创建主题和订阅、发送和接收消息以及监控事件和日志。
- AWS Command Line Interface (AWS CLI) 允许您直接访问 Amazon 以SNSAPI获取高级配置和自动化用例。有关更多信息，请参阅将 Amazon SNS 与 AWS CLI。
- AWS SDKs以多种语言提供。有关更多信息，请参阅SDKs和工具包。

亚马逊的定价 SNS

Amazon SNS 没有预付费用。您根据发布的消息数量、发送的通知数量以及任何其他用于管理主题和订阅的API电话进行付费。传输定价因终端节点类型而异。您可以免费开始使用 Amazon SNS 免费套餐。

有关信息，请参阅 Amazon SNS 定价。

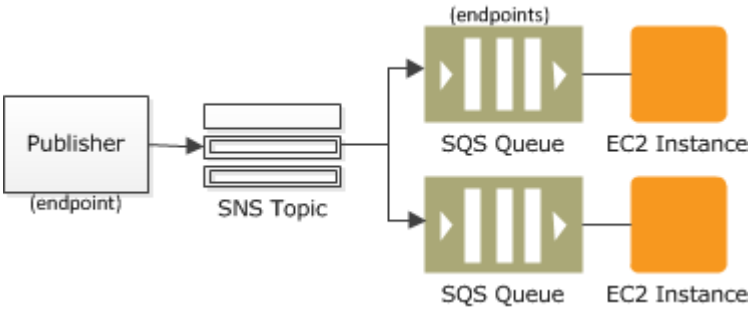
常见的亚马逊SNS场景

了解 Amazon SNS 如何通过将消息复制到多个终端节点、支持订单处理和系统测试等现实应用程序来增强异步处理。

应用程序集成

Fanout 场景是将发布到某个SNS主题的消息复制并推送到多个终端节点，例如 Firehose 交付流、A SQS mazon 队列HTTP、(S) 终端节点和 Lambda 函数。这允许进行并行异步处理。

例如，您可以开发一个应用程序，该应用程序可以在下单产品时向SNS主题发布消息。然后，订阅该SNS主题的SQS队列会收到与新订单相同的通知。连接到其中一个SQS队列的亚马逊弹性计算云 (AmazonEC2) 服务器实例可以处理或配送订单。您还可以将另一个 Amazon EC2 服务器实例附加到数据仓库中，以分析收到的所有订单。



您还可以通过扇出复制使用您的测试环境复制发送到生产环境的数据。在前一个示例的基础上，您可以为另一个SQS队列订阅同一SNS主题，以获取新的传入订单。然后，通过将这个新SQS队列连接到您的测试环境，您可以继续使用从生产环境中接收到的数据来改进和测试您的应用程序。

> **Important**
>
> 在将任何生产数据发送到测试环境之前，请确保您考虑数据隐私和安全性。

有关更多信息，请参阅以下资源：

- Fanout Amazon SNS 通知，带有 Firehose 传送流，可增强数据管理
- Fanout Amazon SNS 向 Lambda 函数发送通知，用于自动处理
- Fanout Amazon SNS 通知到亚马逊SQS队列进行异步处理
- Fanout Amazon SNS 向 HTTP /S 端点发送通知
- 使用 Amazon SNS 以及 AWS 计算、存储、数据库和网络服务实现事件驱动型计算

应用程序提示

应用程序和系统提示是由预定义阈值触发的通知。Amazon SNS 可以通过SMS和电子邮件将这些通知发送给指定用户。例如，当事件发生时，您可以立即收到通知，例如您的 Amazon A EC2 uto Scaling 组的特定更改、上传到 Amazon S3 存储桶的新文件或亚马逊的指标阈值被突破。 CloudWatch有关更多信息，请参阅《亚马逊 CloudWatch 用户指南》中的设置亚马逊SNS通知。

用户通知

Amazon SNS 可以向个人或团体发送推送电子邮件和SMS短信（消息）。例如，您可以将电子商务订单确认作为用户通知发送。有关使用 Amazon SNS 发送SMS消息的更多信息，请参阅通过 Amazon 发送移动短信 SNS。

移动推送通知

使用移动推送通知，可将消息直接推送到移动应用程序。例如，您可以使用 Amazon SNS 向应用程序发送更新通知。通知消息可以包含下载和安装更新的链接。有关使用 Amazon SNS 发送推送通知消息的更多信息，请参阅通过 Amazon 发送移动推送通知 SNS。

将 Amazon SNS 与 AWS SDK

AWS 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个版本都SDK提供了API代码示例和文档，使开发人员可以更轻松地使用自己的首选语言构建应用程序。

SDK文档	代码示例
AWS SDK for C++	AWS SDK for C++ 代码示例
AWS CLI	AWS CLI 代码示例
AWS SDK for Go	AWS SDK for Go 代码示例
AWS SDK for Java	AWS SDK for Java 代码示例
AWS SDK for JavaScript	AWS SDK for JavaScript 代码示例
AWS SDK for Kotlin	AWS SDK for Kotlin 代码示例
AWS SDK for .NET	AWS SDK for .NET 代码示例
AWS SDK for PHP	AWS SDK for PHP 代码示例
AWS Tools for PowerShell	PowerShell 代码示例工具
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 代码示例
AWS SDK for Ruby	AWS SDK for Ruby 代码示例
AWS SDK for Rust	AWS SDK for Rust 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
AWS SDK for Swift	AWS SDK for Swift 代码示例

有关 Amazon 的特定示例SNS，请参阅SNS使用 Amazon 的代码示例 AWS SDKs。

> **示例可用性**
>
> 找不到所需的内容？ 通过使用此页面底部的提供反馈链接请求代码示例。

了解如何创建 Amazon SNS 主题和发布消息

本主题提供了管理 Amazon SNS 资源的基本步骤，特别侧重于主题、订阅和消息发布。首先，您将为亚马逊设置必要的访问权限SNS，确保您拥有创建和管理亚马逊SNS资源的正确权限。接下来，您将创建一个新的 Amazon SNS 主题，该主题将作为管理和向订阅者发送消息的中心中心。创建主题后，您将继续创建对此主题的订阅，允许特定的终端节点接收发布到该主题的消息。

主题和订阅完成后，您将向该主题发布一条消息，观察 Amazon 如何SNS高效地将消息传送到所有已订阅的终端节点。最后，您将学习如何删除订阅和主题，从而完成您管理的 Amazon SNS 资源的生命周期。这种方法可以让您清楚地了解亚马逊的基本操作SNS，使您掌握使用亚马逊SNS控制台管理消息工作流程所需的实用技能。

主题

为 Amazon 设置访问权限 SNS

在首次使用 Amazon SNS 之前，您必须完成以下步骤。

创建一个 AWS 账户 和一个IAM用户

要访问任何 AWS 服务，必须先创建一个AWS 账户。您可以使用 AWS 账户 来查看您的活动和使用情况报告，以及管理身份验证和访问权限。

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。

2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，将创建一个 AWS 账户 root 用户。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后 AWS 账户，请保护您的 AWS 账户 root 用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护您的 AWS 账户 root 用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的 root 用户开启多重身份验证 (MFA)。

   有关说明，请参阅《用户指南》中的 “为 AWS 账户 root 用户（控制台）启用虚拟MFA设备” IAM。

创建具有管理访问权限的用户

1. 启用IAM身份中心。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，向用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 Ident IAM ity Center 用户登录URL，请使用您在创建 Ident IAM ity Center 用户时发送到您的电子邮件地址的登录信息。

  有关使用 Ident IAM ity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个遵循应用最低权限权限的最佳实践的权限集。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

创建亚马逊SNS主题

Amazon SNS 主题是充当通信渠道的逻辑接入点。主题允许您对多个终端节点（例如 Amazon AWS Lambda SQS、HTTP /S 或电子邮件地址）进行分组。

要广播使用需要其消息的多个其他服务（例如，结算和执行系统）的消息创建器系统（例如，电子商务网站）的消息，您可以为创建器系统创建主题。

Amazon 的第一个也是最常见的SNS任务是创建主题。本页显示如何使用 AWS Management Console AWS SDK for Java、和 AWS SDK for .NET ，创建主题。

在创建过程中，您可以选择主题类型（标准或FIFO）并命名该主题。创建主题后，无法更改主题类型或名称。在创建主题期间，所有其他配置选项都是可选的，您可以稍后对其进行编辑。

> **Important**
>
> 请勿在主题名称中添加个人身份信息 (PII) 或其他机密或敏感信息。其他 Amazon Web Services 可以访问主题名称，包括 CloudWatch 日志。主题名称不适合用于私有或敏感数据。

主题

要使用创建主题 AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 请执行以下操作之一：
   - 如果 AWS 账户 之前未在您下创建过任何主题，请阅读主页SNS上的 Amazon 描述。
   - 如果 AWS 账户 之前已在您的下方创建过主题，请在导航面板上选择主题。
3. 在 Topics（主页）页面上，选择 Create topic（创建主题）。
4. 在 Create topic（创建主题）页面上，在 Details（详细信息）部分中，执行以下操作：
   a. 在 “类型” 中，选择主题类型（标准或 FIFO）。
   b. 输入主题的名称。对于FIFO主题，请在名称末尾添加 .fifo。
   c. （可选）输入主题的显示名称。

      > **Important**
      >
      > 订阅电子邮件终端节点时，亚马逊SNS主题显示名称和发送电子邮件地址（例如 no-reply@sns.amazonaws.com）的组合字符数不得超过 320 UTF -8 个字符。在为 Amazon SNS 主题配置显示名称之前，您可以使用第三方编码工具来验证发送地址的长度。

   d. （可选）对于FIFO主题，您可以选择基于内容的消息重复数据删除来启用默认的消息重复数据删除。有关更多信息，请参阅 针对FIFO主题的 Amazon SNS 消息重复数据删除。
5. （可选）展开加密部分并执行以下操作。有关更多信息，请参阅 使用服务器端加密保护 Amazon SNS 数据。
   a. 选择 Enable encryption（启用加密）。
   b. 指定密 AWS KMS 钥。有关更多信息，请参阅 关键术语。

      对于每KMS种类型，都会显示描述、账户KMSARN和。

> **Important**
>
> 如果您不是的所有者KMS，或者您使用不具有`kms:ListAliases`和`kms:DescribeKey`权限的账户登录，则将无法在 Amazon SNS 控制台KMS上查看相关信息。要求的所有者授KMS予您这些权限。有关更多信息，请参阅《AWS Key Management Service 开发者指南》中的 “AWS KMS API权限：操作和资源参考”。

- 默认选择亚马逊 AWS 托管 KMSSNS（默认）别名/aws/sns。

  > **Note**
  >
  > 记住以下内容：
  >
  > - 首次使用KMS为主题指定亚马逊 AWS 托管时， AWS KMS 会创建SNSKMS适用于亚马逊的 AWS 托管SNS。 AWS Management Console
  > - 或者，在您首次对SSE启用状态的主题使用`Publish`操作时， AWS KMS 会创建KMS适用于 Amazon 的 AWS 托管SNS。

- 要使用 AWS 账户KMS中的自定义项，请选择KMS关键字段，然后KMS从列表中选择自定义字段。

  > **Note**
  >
  > 有关创建自定义密钥的说明KMSs，请参阅《AWS Key Management Service 开发者指南》中的创建密钥

- 要使用您 AWS 账户或其他账户KMSARN中的自定义设置，请在KMS密钥字段中输入该自定义项。 AWS

6. （可选）默认情况下，只有主题拥有者才能发布或订阅主题。要配置其他访问权限，请展开访问策略部分。有关更多信息，请参阅Amazon 中的身份和访问管理 SNS 和Amazon SNS 访问控制的示例案例。

> **Note**
>
> 使用控制台创建主题时，默认策略使用 `aws:SourceOwner` 条件键。此密钥类似于 `aws:SourceAccount`。

7. （可选）要配置 Amazon 如何SNS重试失败的消息传送尝试，请展开传送重试策略 (HTTP/S) 部分。有关更多信息，请参阅 Amazon SNS 消息传送重试次数。
8. （可选）要配置 Amazon 如何SNS记录向其发送的消息 CloudWatch，请展开 “传送状态记录” 部分。有关更多信息，请参阅 Amazon SNS 消息传送状态。
9. （可选）要将元数据标签添加到主题中，请展开标签部分，输入一个键和值（可选），然后选择添加标签。有关更多信息，请参阅 Amazon SNS 话题标记。
10. 选择创建主题。

    主题已创建，然后 ***MyTopic***页面已显示。

    主题的名称ARN、（可选）显示名称和主题所有者的 AWS 账户 ID 显示在 “详细信息” 部分中。
11. 将主题复制ARN到剪贴板，例如：

```
arn:aws:sns:us-east-2:123456789012:MyTopic
```

使用创建主题 AWS SDK

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置AWS SDKs和凭据文件。

以下代码示例演示如何使用 `CreateTopic`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用特定的名称创建主题。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
```

```
    }
```

创建一个具有名称、特定属FIFO性和重复数据消除属性的新主题。

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }
```

```
        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 CreateTopic” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
```

```
                << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 CreateTopic” 中的。

CLI

AWS CLI

创建 SNS 主题

以下create-topic示例创建了一个名为SNS的主题my-topic。

```
aws sns create-topic \
    --name my-topic
```

输出：

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

有关更多信息，请参阅《 AWS 命令行界面用户指南》SNS中的在 Amazon SQS 和 Amazon 上使用AWS命令行界面。

- 有关API详细信息，请参阅AWS CLI 命令参考CreateTopic中的。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
```

```
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- 有关API详细信息，请参阅 “AWS SDK for Go API参考 CreateTopic” 中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
```

```
                    topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 CreateTopic” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
```

```
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 CreateTopic” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- 有关API详细信息，请参阅CreateTopic中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 CreateTopic” 中的。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- 有关API详细信息，请参阅CreateTopic中的 AWS SDKPython (Boto3) API 参考。

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# This class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
```

```
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for Ruby API参考 CreateTopic” 中的。

Rust

SDK对于 Rust

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- 有关API详细信息，请参见CreateTopic中的 Rust AWS SDK API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   TRY.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
is returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- 有关API详细信息，请参阅CreateTopic中的AWS SDK以供SAPABAPAPI参考。

创建对 Amazon SNS 主题的订阅

要接收发布至某个主题的消息，您必须订阅一个到该主题的终端节点。在为终端节点订阅主题后，此终端节点会开始接收发布到关联主题的消息。

> **Note**
>
> HTTP(S) 端点、电子邮件地址和其他 AWS 资源 AWS 账户 需要在订阅确认后才能接收消息。

为终端节点订阅亚马逊SNS主题

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择订阅。
3. 在 Subscriptions（订阅）页面上，选择 Create subscription（创建订阅）。
4. 在 Create subscription（创建订阅）页上的 Details（详细信息）部分中，执行以下操作：

a. 对于主题 ARN，选择主题的 Amazon 资源名称 (ARN)。例如 AWS ARN，此值就是您在创建 Amazon SNS 主题时生成的值`arn:aws:sns:us-east-2:123456789012:your_topic`。

b. 对于 Protocol（协议），选择终端节点类型。可用的终端节点类型包括：

- HTTP/HTTPS
- 电子邮件/电子邮件-JSON
- Amazon Data Firehose
- 亚马逊 SQS

> **Note**
>
> 要订阅某个SNSFIFO主题，请选择此选项。

- AWS Lambda
- 平台应用程序终端节点
- SMS

c. 对于 En d point，输入终端节点值，例如电子邮件地址或 Amazon SQS 队列的。ARN

d. 仅限 Firehose 端点：对于订阅角色 ARN，请指定您为写入 Firehose 交付流而创建的IAM角色。ARN有关更多信息，请参阅 将 Firehose 直播订阅亚马逊主题的先决条件 SNS。

e. （可选）对于 Firehose、Amazon SQS、HTTP /S 终端节点，您还可以启用原始消息传输。有关更多信息，请参阅 Amazon SNS 原始消息传送。

f. （可选）要配置筛选策略，请展开 Subscription filter policy（订阅筛选策略）部分。有关更多信息，请参阅 亚马逊SNS订阅筛选政策。

g. （可选）要启用基于有效负载的筛选，请将 `Filter Policy Scope` 配置为 `MessageBody`。有关更多信息，请参阅 Amazon SNS 订阅筛选政策范围。

h. （可选）要为订阅配置死信队列，请展开 Redrive policy (dead-letter queue)（重新驱动策略（死信队列））部分。有关更多信息，请参阅 Amazon SNS 死信队列。

i. 选择创建订阅。

控制台将创建订阅并打开订阅的 Details（详细信息）页面。

向 Amazon SNS 主题发布消息

创建 Amazon SNS 主题并订阅该主题的终端节点后，您可以向该主题发布消息。消息发布后，Amazon SNS 会尝试将消息传送到已订阅的终端节点。

主题

- 要使用 Amazon SNS 主题发布消息 AWS Management Console
- 使用向主题发布消息 AWS SDK
- 使用 Amazon SNS 和 Amazon S3 发布大消息
- Amazon SNS 消息属性
- 亚马逊SNS消息批处理

要使用 Amazon SNS 主题发布消息 AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择主题。
3. 在 Topics（主题）页上，选择一个主题，然后选择Publish message（发布主题）。

   控制台将打开 Publish message to topic（将消息发布到主题）页面。
4. 在 Message details（消息详细信息）部分中，执行以下操作：

   a. （可选）输入消息 Subject（主题）。

   b. 对于FIFO主题，请输入消息组 ID。同一消息组中的消息按消息的发布顺序传输。

   c. 对于FIFO主题，请输入消息重复数据删除 ID。如果您为主题启用了 Content-based message deduplication（基于内容的消息重复数据删除）设置，则此 ID 为可选项。

   d. （可选）对于移动推送通知，请输入以秒为单位的生存时间 (TTL) 值。这是推送通知服务（例如 Apple 推送通知服务 (APNs) 或 Firebase Cloud Messaging (FCM)）将消息传递到端点所需的时间。
5. 在 Message body (消息正文) 部分中，执行以下操作之一：

   a. 选择 Identical payload for all delivery protocols（完全相同负载用于所有交付协议），然后输入消息。

   b. 为每个传输协议选择 “自定义负载”，然后输入一个JSON对象来定义要为每个传输协议发送的消息。

有关更多信息，请参阅 发布带有特定平台有效负载的 Amazon SNS 通知。

6. 在消息属性部分，添加您希望 Amazon SNS 与订阅属性匹配的所有属性，FilterPolicy以决定订阅的终端节点是否对已发布的消息感兴趣。

   a. 对于 Type（类型），选择属性类型，例如 String.Array。

   > **Note**
   >
   > 对于属性类型 String.Array，请将该数组放入方括号 ([]) 内。在该数组内，将字符串值加入双引号内。数字以及关键字 true、false 和 null 无需加引号。

   b. 输入属性名称，例如 customer_interests。

   c. 输入属性值，例如 ["soccer", "rugby", "hockey"]。

   如果属性类型为 Strin g、String.Array 或数字，则在未明确设置筛选策略范围的情况下，Amazon 会根据订阅的筛选策略（如果存在）SNS评估消息属性，然后再将消息发送到订阅。MessageBody

   有关更多信息，请参阅 Amazon SNS 消息属性。

7. 选择发布消息。

   消息将发布到主题，且控制台将打开主题的 Details（详细信息）页面。

使用向主题发布消息 AWS SDK

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置AWS SDKs和凭据文件。

以下代码示例演示如何使用 Publish。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

向主题发布消息。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
```

```
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

使用组、复制和属性选项向主题发布消息。

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
```

```
                                    "\r\nAll messages within the same group will be
received in the order " +
                                    "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }
```

```
            keepSendingMessages = GetYesNoResponse("Send another message?",
 false);
        }
    }
```

将用户的选择应用于发布操作。

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
```

```
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- 有关API详细信息，请参阅在AWS SDK for .NET API参考中发布。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

发布带有属性的消息。

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
```

```
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- 有关API详细信息，请参阅在AWS SDK for C++ API参考中发布。

CLI

AWS CLI

示例 1：向主题发布消息

以下publish示例将指定的消息发布到指定的SNS主题。该消息来自一个文本文件，您可以在该文件中包含换行符。

```
aws sns publish \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
    --message file://message.txt
```

message.txt 的内容：

```
Hello World
```

```
Second Line
```

输出：

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

示例 2：向电话号码发布SMS消息

以下 publish 示例将消息 Hello world! 发布到电话号码 +1-555-555-0100。

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

输出：

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- 有关API详细信息，请参阅在《AWS CLI 命令参考》中发布。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
```

```
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- 有关API详细信息，请参阅在AWS SDK for Go API参考中发布。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
```

```
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅在AWS SDK for Java 2.x API参考中发布。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                    if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
```

```
  return response;
};
```

使用组、复制和属性选项向主题发布消息。

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
```

```
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅在AWS SDK for JavaScript API参考中发布。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun pubTopic(
    topicArnVal: String,
```

```
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- 有关API详细信息，请参阅发布AWS SDK以获取 Kotlin API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
```

```
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅在AWS SDK for PHP API参考中发布。

PowerShell

用于 PowerShell

示例 1：此示例显示发布一条 MessageAttribute 声明为内联的消息。

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

示例 2：此示例显示发布一条事先 MessageAttributes 声明了多条消息的消息。

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"
```

```
$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- 有关API详细信息，请参阅在 AWS Tools for PowerShell Cmdlet 参考中发布。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

发布包含属性的消息，以便订阅可以根据属性进行筛选。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
```

```
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message.
 Values
                            must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

发布基于订阅者的协议采取不同形式的消息。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource
```

```
    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
        :param default_message: The default version of the message. This version
is
                                sent to subscribers that have protocols that are
not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

- 有关API详细信息，请参阅 Python 中发布AWS SDK以供参考 (Boto3) API。

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"         # Should be replaced with the actual message
 content
```

```
  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅在AWS SDK for Ruby API参考中发布。

Rust

SDK对于 Rust

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);
```

```
    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- 有关API详细信息，请参见发布AWSSDK以供 Rust API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message
        ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅发布AWSSDK以供SAPABAPAPI参考。

使用 Amazon SNS 和 Amazon S3 发布大消息

要发布大型亚马逊SNS消息，您可以使用适用于 Java 的亚马逊SNS扩展客户端库或适用于 Python 的亚马逊SNS扩展客户端库。对于大于当前最大值 256KB（最大为 2GB）的消息，这些库非常有用。这两个库都将实际负载保存到 Amazon S3 存储桶中，并将存储的 Amazon S3 对象的引用发布到亚马逊SNS主题。已订阅的亚马逊SQS队列可以使用适用于 Java 的亚马逊SQS扩展客户端库来取消引用并检索来自 Amazon S3 的有效负载。其他端点（如 Lambda）可以使用 AWS的有效负载卸载 Java 公共库来取消引用并检索有效负载。

> **Note**
>
> Amazon SNS 扩展客户端库与标准版和FIFO主题兼容。

主题

适用于 Java 的亚马逊 Java SNS 扩展客户端库

主题

先决条件

以下是使用适用于 Java 的 Amazon SNS 扩展客户端库的先决条件：

- 一个 AWS SDK。

  本页上的示例使用 AWS Java SDK。要安装和设置SDK，请参阅《AWS SDK for Java 开发人员指南》中的“ AWS SDK为 Java 设置”。
- 并 AWS 账户 具有正确的凭据。

  要创建 AWS 账户，请导航到AWS 主页，然后选择创建 AWS 帐户。按照说明进行操作。

有关证书的信息，请参阅《AWS SDK for Java 开发人员指南》中的设置 AWS 证书和开发区域。

- Java 8 或更高版本。
- 适用于 Java 的亚马逊SNS扩展客户端库（也可从 Maven 获得）。

配置消息存储

Amazon SNS 扩展客户端库使用负载卸载 Java 公共库 AWS 进行消息存储和检索。您可以配置以下 Amazon S3 消息存储选项：

- 自定义消息大小阈值 – 具有超过此大小的负载和属性的消息将自动存储在 Amazon S3 中。
- `alwaysThroughS3` 标志 – 将此值设置为 `true` 以强制将所有消息负载存储在 Amazon S3 中。例如：

```
SNSExtendedClientConfiguration snsExtendedClientConfiguration = new
SNSExtendedClientConfiguration() .withPayloadSupportEnabled(s3Client,
 BUCKET_NAME).withAlwaysThroughS3(true);
```

- 自定义KMS密钥-用于在 Amazon S3 存储桶中进行服务器端加密的密钥。
- 存储桶名称 – 用于存储消息负载的 Amazon S3 存储桶的名称。

示例：使用存储在 Amazon SNS S3 中的有效负载向亚马逊发布消息

以下代码示例展示了如何：

- 创建示例主题和队列。
- 订阅队列以接收来自主题的消息。
- 发布测试消息。

消息负载存储在 Amazon S3，以及发布到的引用中。Amazon SQS 扩展客户端用于接收消息。

SDK适用于 Java 1.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

要发布大型消息，请使用适用于 Java 的 Amazon SNS 扩展客户端库。您发送的消息将引用包含实际消息内容的 Amazon S3 对象。

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {

        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;

                // Message threshold controls the maximum message size that will be
 allowed to
                // be published
                // through SNS using the extended client. Payload of messages
 exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
 maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
```

```
                final AmazonSNS snsClient =
AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
AmazonSQSClientBuilder.standard().withRegion(region).build();
                final AmazonS3 s3Client =
AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
                final String topicArn = snsClient.createTopic(
                                new
CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);

                // To read message content stored in S3 transparently through SQS
extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
                final SetSubscriptionAttributesRequest subscriptionAttributesRequest
= new SetSubscriptionAttributesRequest();
                subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");
                snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);

                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3 when
the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
                final SNSExtendedClientConfiguration snsExtendedClientConfiguration
= new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

.withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);
```

```
                // Publish message via SNS with storage in S3
                final String message = "This message is stored in S3 as it exceeds
 the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
                final ExtendedClientConfiguration sqsExtendedClientConfiguration =
 new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
 AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
 sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
 result.getMessages().get(0).getBody());
        }
}
```

其他终端节点协议

Amazon SNS 和 Amazon SQS 库都使用负载卸载 Java 公共库 AWS来存储和检索带有 Amazon S3 的消息负载。任何支持 Java 的端点（例如，在 Java 中实现的HTTPS端点）都可以使用同一个库来取消引用消息内容。

无法使用有效负载卸载 Java 公共库的终端节点仍然 AWS 可以发布存储在 Amazon S3 中的有效负载的消息。以下是由上面的代码示例发布的 Amazon S3 引用的示例：

```
[
  "software.amazon.payloadoffloading.PayloadS3Pointer",
  {
    "s3BucketName": "extended-client-bucket",
    "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
  }
]
```

适用于 Python 的亚马逊SNS扩展客户端库

主题

- 先决条件
- 配置消息存储
- 示例：使用存储在 Amazon SNS S3 中的有效负载向亚马逊发布消息

先决条件

以下是使用适用于 Python 的亚马逊SNS扩展客户端库的先决条件：

- 一个 AWS SDK。

  本页上的示例使用 AWS Python SDK Boto3。要安装和设置SDK，请参阅 Python 文档。AWS SDK
- 并 AWS 账户 具有正确的凭据。

  要创建 AWS 账户，请导航到AWS 主页，然后选择创建 AWS 帐户。按照说明进行操作。

  有关证书的信息，请参阅 Python 开发者指南中的凭证。AWS SDK
- Python 3.x（或更高版本）和 pip。
- 适用于 Python 的亚马逊SNS扩展客户端库（也可从 PyPI 获得）。

配置消息存储

Boto3 Amazon SNS 客户端、主题和PlatformEndpoint对象上提供了以下属性，用于配置 Amazon S3 消息存储选项。

- `large_payload_support` – 用于存储大型消息的 Amazon S3 桶名称。
- `message_size_threshold` – 在大型消息桶中存储消息的阈值。不能低于 0 或超过 262144。原定设置值为 262144。
- `always_through_s3` – 如果为 `True`，则所有消息都存储在 Amazon S3 中。默认为 `False`。
- `s3` – 用于在 Amazon S3 中存储对象的 Boto3 Amazon S3 `resource` 对象。如果您想要控制 Amazon S3 资源（例如，自定义 Amazon S3 配置或凭证），请使用此选项。如果之前在首次使用时未设置，则原定设置值为 `boto3.resource("s3")`。

示例：使用存储在 Amazon SNS S3 中的有效负载向亚马逊发布消息

以下代码示例展示了如何：

- 创建示例亚马逊SNS主题和亚马逊SQS队列。

- 订阅队列以接收来自主题的消息。
- 发布测试消息。
- 消息有效负载存储在 Amazon S3 中，并发布对它的引用。
- 打印队列中已发布的消息以及从 Amazon S3 检索到的原始消息。

要发布大型消息，请使用适用于 Python 的亚马逊SNS扩展客户端库。您发送的消息将引用包含实际消息内容的 Amazon S3 对象。

```
import boto3
import sns_extended_client
from json import loads

s3_extended_payload_bucket = "extended-client-bucket-store"
TOPIC_NAME = "TOPIC-NAME"
QUEUE_NAME = "QUEUE-NAME"

# Create an helper to fetch message from S3
def get_msg_from_s3(body):
    json_msg = loads(body)
    s3_client = boto3.client("s3")
    s3_object = s3_client.get_object(
        Bucket=json_msg[1].get("s3BucketName"), Key=json_msg[1].get("s3Key")
    )
    msg = s3_object.get("Body").read().decode()
    return msg

# Create an helper to fetch and print message SQS queue and S3
def fetch_and_print_from_sqs(sqs, queue_url):
    """Handy Helper to fetch and print message from SQS queue and S3"""
    message = sqs.receive_message(
        QueueUrl=queue_url, MessageAttributeNames=["All"], MaxNumberOfMessages=1
    ).get("Messages")[0]
    message_body = message.get("Body")
    print("Published Message: {}".format(message_body))
    print("Message Stored in S3 Bucket is: {}\n".format(get_msg_from_s3(message_body)))

# Initialize the SNS client and create SNS Topic
sns_extended_client = boto3.client("sns", region_name="us-east-1")
create_topic_response = sns_extended_client.create_topic(Name=TOPIC_NAME)
demo_topic_arn = create_topic_response.get("TopicArn")
```

```
# Create and subscribe an SQS queue to the SNS topic
sqs = boto3.client("sqs")
demo_queue_url = sqs.create_queue(QueueName=QUEUE_NAME).get("QueueUrl")
demo_queue_arn = sqs.get_queue_attributes(QueueUrl=demo_queue_url,
AttributeNames=["QueueArn"])["Attributes"].get("QueueArn")
# Set the RawMessageDelivery subscription attribute to TRUE
sns_extended_client.subscribe(TopicArn=demo_topic_arn, Protocol="sqs",
Endpoint=demo_queue_arn, Attributes={"RawMessageDelivery":"true"})

sns_extended_client.large_payload_support = s3_extended_payload_bucket

# To store all messages content in S3, set always_through_s3 to True
# In the example, we set message size threshold as 32 bytes, adjust this threshold as
 per your usecase
# Message will only be uploaded to S3 when its payload size exceeded threshold
sns_extended_client.message_size_threshold = 32
sns_extended_client.publish(
    TopicArn=demo_topic_arn,
    Message="This message should be published to S3 as it exceeds the
 message_size_threshold limit",
)
# Print message stored in s3
fetch_and_print_from_sqs(sqs, demo_queue_url)
```

输出

```
Published Message:
[
    "software.amazon.payloadoffloading.PayloadS3Pointer",
    {
        "s3BucketName": "extended-client-bucket-store",
        "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
    }
]
Message Stored in S3 Bucket is: This message should be published to S3 as it exceeds
 the message_size_threshold limit
```

Amazon SNS 消息属性

Amazon SNS 支持发送消息属性，允许您提供有关消息的结构化元数据项目（例如时间戳、地理空间数据、签名和标识符）。对于SQS订阅，启用原始消息传送后，最多可以发送 10 个消息属性。要发送

10 个以上的消息属性，必须禁用 Raw Message Delivery（原始消息传输）。定向到启用了原始消息传送功能的 Amazon SQS 订阅的 10 个以上消息属性的消息将作为客户端错误而被丢弃。

消息属性是可选的，并独立于消息正文（但随之一起发送）。接收方可以使用此信息来决定如何处理消息，而不必先处理消息正文。

有关使用 AWS Management Console 或发送带有属性的消息的信息 AWS SDK for Java，请参阅要使用 Amazon SNS 主题发布消息 AWS Management Console教程。

> **Note**
>
> 只有当消息结构为字符串时，才会发送消息属性，而不是JSON。

您还可以使用消息属性，帮助构造移动终端节点的推送通知消息。在这种情况下，消息属性仅用于帮助构造推送通知消息。这些属性不会像向亚马逊终端节点发送带有消息属性的消息时那样传送到SQS终端节点。

您还可以使用消息属性来让消息变为可通过订阅筛选策略进行筛选。可以将筛选策略应用于主题订阅。应用了筛选策略且筛选策略范围设置为 `MessageAttributes`（默认值）时，订阅将只接收具有策略接受的属性的那些消息。有关更多信息，请参阅 Amazon SNS 邮件过滤。

> **Note**
>
> 当使用消息属性进行筛选时，该值必须是有效的JSON字符串。这样做可以确保将消息传送到启用了消息属性筛选的订阅。

消息属性项目和验证

每个消息属性包含以下项目：

- Name – 消息属性的名称可以包含以下字符：A-Z、a-z、0-9、下划线 (_)、连字符 (-) 和句点 (.)。名称不得以句点开头或结尾，并且不应包含连续句点。名称区分大小写，且必须在消息的所有属性名称中是唯一的。名称最多可以有 256 个字符。名称不能以 `AWS.` 或 `Amazon.`（或任何大小写变化形式）开头，因为这些前缀已预留以供 Amazon Web Services 使用。
- Type – 受支持的消息属性数据类型有 `String`、`String.Array`、`Number` 和 `Binary`。数据类型在内容方面具有与消息正文相同的限制。数据类型区分大小写，长度最多可以为 256 字节。想要了解更多信息，请参阅消息属性数据类型和验证部分。

- Value – 用户指定的消息属性值。对于字符串数据类型，值属性在内容方面具有与消息正文相同的限制。有关更多信息，请参阅《亚马逊简单通知服务API参考》中的 “发布” 操作。

名称、类型和值都不得为空或 null。此外，消息正文也不应为空或 null。消息属性的所有部分 (包括名称、类型和值) 都包含在消息大小限制中，该限制当前是 256 KB。

消息属性数据类型和验证

消息属性数据类型用于确定 Amazon 如何处理消息属性值SNS。例如，如果类型为数字，Amazon SNS 会验证它是否为数字。

除另有说明外，Amazon SNS 支持所有终端节点的以下逻辑数据类型：

- 字符串 — 字符串是 Unicode，二进制编码为 UTF -8。有关代码值的列表，请参阅 http://en.wikipedia.org/wiki/ ASCII # ASCII _printable_ characters。

  > **Note**
  >
  > 消息属性中不支持代理值。例如，使用代理值来表示表情符号将收到以下错误：`Invalid attribute value was passed in for message attribute`。

- String.Array – 格式为字符串的阵列，可以包含多个值。这些值可以是字符串、数字或关键字 `true`、`false` 和 `null`。数字或布尔类型的 String.Array 不需要引号。多个 String.Array 值用逗号分隔。

  AWS Lambda 订阅不支持此数据类型。如果您为 Lambda 终端节点指定此数据类型，则该数据类型将作为`String`数据类型传递到亚马逊SNS传输给 Lambda 的JSON有效负载中。

- Number – 数字是正或负整数或是浮点数。数字具有足够的范围和精度，以便包含整数、浮点数和双精度数通常支持的大多数可能值。数字的值可以介于 -10^9 到 10^9 之间，精确至小数点后 5 位数。系统会删减开头和结尾的 0。

  AWS Lambda 订阅不支持此数据类型。如果您为 Lambda 终端节点指定此数据类型，则该数据类型将作为`String`数据类型传递到亚马逊SNS传输给 Lambda 的JSON有效负载中。

- Binary – 二进制类型属性可以存储任何二进制数据，例如压缩数据、加密数据或图像。

为移动推送通知预留的消息属性

下表列出了可用于构造推送通知消息的移动推送通知服务的预留消息属性：

推送通知服务	预留的消息属性
ADM	AWS.SNS.MOBILE.ADM.TTL
APNs[1]	AWS.SNS.MOBILE.APNS_MDM.TTL
	AWS.SNS.MOBILE.APNS_MDM_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS_PASSBOOK.TTL
	AWS.SNS.MOBILE.APNS_PASSBOOK_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS_VOIP.TTL
	AWS.SNS.MOBILE.APNS_VOIP_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS.COLLAPSE_ID
	AWS.SNS.MOBILE.APNS.PRIORITY
	AWS.SNS.MOBILE.APNS.PUSH_TYPE
	AWS.SNS.MOBILE.APNS.TOPIC
	AWS.SNS.MOBILE.APNS.TTL
Baidu	AWS.SNS.MOBILE.BAIDU.DeployStatus
	AWS.SNS.MOBILE.BAIDU.MessageKey
	AWS.SNS.MOBILE.BAIDU.MessageType
	AWS.SNS.MOBILE.BAIDU.TTL
FCM	AWS.SNS.MOBILE.FCM.TTL
	AWS.SNS.MOBILE.GCM.TTL
macOS	AWS.SNS.MOBILE.MACOS_SANDBOX.TTL

推送通知服务	预留的消息属性
	AWS.SNS.MOBILE.MACOS.TTL
MPNS	AWS.SNS.MOBILE.MPNS.NotificationClass
	AWS.SNS.MOBILE.MPNS.TTL
	AWS.SNS.MOBILE.MPNS.Type
WNS	AWS.SNS.MOBILE.WNS.CachePolicy
	AWS.SNS.MOBILE.WNS.Group
	AWS.SNS.MOBILE.WNS.Match
	AWS.SNS.MOBILE.WNS.SuppressPopup
	AWS.SNS.MOBILE.WNS.Tag
	AWS.SNS.MOBILE.WNS.TTL
	AWS.SNS.MOBILE.WNS.Type

[1] 如果消息属性不符合其要求，Apple 将拒绝亚马逊的SNS通知。如需了解更多详情，请参阅 Apple 开发者网站上的 “向发送通知请求”。APNs

亚马逊SNS消息批处理

什么是消息批处理？

除了向标准版或单个PublishAPI请求中的FIFO主题发布消息之外，另一种方法是使用 Amazon SNS PublishBatch API 在单个API请求中发布最多 10 条消息。批量发送消息可以帮助您将连接分布式应用程序（A2A 消息）或通过 A ma SNS zon 向他人发送通知（A2P 消息）相关的成本降低多达 10 倍。根据您所在SNS的区域，Amazon 对您每秒可以向一个主题发布多少条消息有配额。有关配额的更多信息，请参阅AWS 一般参考指南中的 Amazon SNS 终端节点和API配额页面。

> **Note**
>
> 您在单个PublishBatchAPI请求中发送的所有消息的总大小不能超过 262,144 字节 (256 KB)。
> 对IAM策略PublishBatchAPI使用相同的PublishAPI操作。

消息批处理是如何工作的？

使用发布消息PublishBatchAPI与使用发布消息类似PublishAPI。主要区别在于，PublishBatchAPI请求中的每条消息都需要分配一个唯一的批处理 ID（最多 80 个字符）。这样，Amazon SNS 就可以为批次中的每条消息返回单独的回API复，以确认每条消息要么已发布，要么发生了故障。对于要发布到FIFO主题的消息，除了包括分配唯一的批处理 ID 之外，您还需要MessageGroupId为每条消息添加MessageDeduplicationID和。

示例

向标准主题发布一批 10 条的消息

```
// Imports
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i))
                .collect(Collectors.toList());
```

```
        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
            System.out.println("Batch Id for successful message: " +
 publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
 publishBatchResultEntry.getMessageId());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
 batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
 batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
 batchResultErrorEntry.getSenderFault());
            System.out.println("Failure Message for failed message: " +
 batchResultErrorEntry.getMessage());
        });
    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

向一个FIFO主题批量发布 10 条消息

```
// Imports
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
```

```
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToFifoTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i)
                        .withMessageGroupId("groupId")
                        .withMessageDeduplicationId("deduplicationId" + i))
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
            System.out.println("Batch Id for successful message: " +
 publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
 publishBatchResultEntry.getMessageId());
            System.out.println("SequenceNumber for successful message: " +
 publishBatchResultEntry.getSequenceNumber());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
 batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
 batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
 batchResultErrorEntry.getSenderFault());
```

```
            System.out.println("Failure Message for failed message: " +
 batchResultErrorEntry.getMessage());
        });

    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

后续步骤

现在，您已经使用订阅创建了一个主题，并向该主题发送了消息，您可能希望尝试以下操作：

- 浏览 AWS 开发人员中心。
- 在 Security（安全性）部分中了解如何保护您的数据。
- 为主题启用服务器端加密。
- 在订阅了加密的亚马逊简单队列服务 (AmazonSQS) 队列的情况下，为主题启用服务器端加密。
- 将 AWS Event Fork Pipelines 订阅到主题。

删除 Amazon SNS 主题和订阅

删除主题后，其关联订阅会异步删除。虽然客户仍然可以访问这些订阅，但即使您使用相同的名称重新创建主题，这些订阅也不再与该主题相关联。

如果发布者尝试向已删除的主题发布消息，则发布者将收到一条错误消息，指出该主题不存在。同样，任何订阅已删除主题的尝试也会导致错误消息。

您无法删除正在等待确认的订阅。Amazon SNS 会在 48 小时后自动删除未经确认的订阅。

主题

- 要删除亚马逊SNS主题或订阅，请使用 AWS Management Console
- 使用删除订阅和主题 AWS SDK

要删除亚马逊SNS主题或订阅，请使用 AWS Management Console

要删除主题，请使用 AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择主题。
3. 在 Topics（主题）页面上，选择一个主题，然后选择 Delete（删除）。
4. 在 Delete topic（删除主题）对话框中，输入 `delete me`，然后选择 Delete（删除）。

   控制台将删除主题。

要删除订阅，请使用 AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择订阅。
3. 在 “订阅” 页面上，选择状态为 “已确认” 的订阅，然后选择 “删除”。
4. 在 Delete subscription（删除订阅）对话框中，选择 Delete（删除）。

   控制台删除订阅。

使用删除订阅和主题 AWS SDK

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置AWS SDKs和凭据文件。

以下代码示例演示如何使用 `DeleteTopic`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

按主题删除主题ARN。

```
    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 DeleteTopic” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);
```

```
    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 DeleteTopic” 中的。

CLI

AWS CLI

删除话SNS题

以下delete-topic示例删除了指定的SNS主题。

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

此命令不生成任何输出。

- 有关API详细信息，请参阅 “DeleteTopic AWS CLI命令参考”。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- 有关API详细信息，请参阅 “AWS SDK for Go API参考 DeleteTopic” 中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 DeleteTopic” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 DeleteTopic” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- 有关API详细信息，请参阅DeleteTopic中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 DeleteTopic” 中的。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- 有关API详细信息，请参阅DeleteTopic中的 AWS SDKPython (Boto3) API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
TRY.
    lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
    MESSAGE 'SNS topic deleted.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关API详细信息，请参阅DeleteTopic中的AWS SDK以供SAPABAPAPI参考。

使用 Amazon SNS FIFO 主题进行消息排序和重复数据删除策略

本主题提供有关 Amazon SNSFIFO（先入先出）主题的特性和功能以及它们如何与 Amazon 队列集成的信息。SQS FIFO您将学习如何将这些服务结合使用，以确保严格的消息排序和重复数据删除，这对于需要数据一致性的应用程序来说是必不可少的。本内容涵盖了 Amazon SNS FIFO 主题有益的特定用例，提供了对消息顺序和唯一性至关重要的场景的见解。

您还将了解消息排序、消息分组的技术细节，以及它们如何影响消息传送。消息重复数据删除主题说明了防止重复消息的机制，确保每条消息只处理一次。此外，你还将学习消息过滤、安全性和耐久性，这些对于维护邮件系统的完整性和可靠性非常重要。此内容还包括有关邮件存档和重播的信息，提供了管理邮件历史记录的策略。还提供了实用的代码示例，以帮助您在自己的应用程序中实现这些功能，让您亲身体验亚马逊SNSFIFO主题及其与亚马逊SQSFIFO队列的集成。

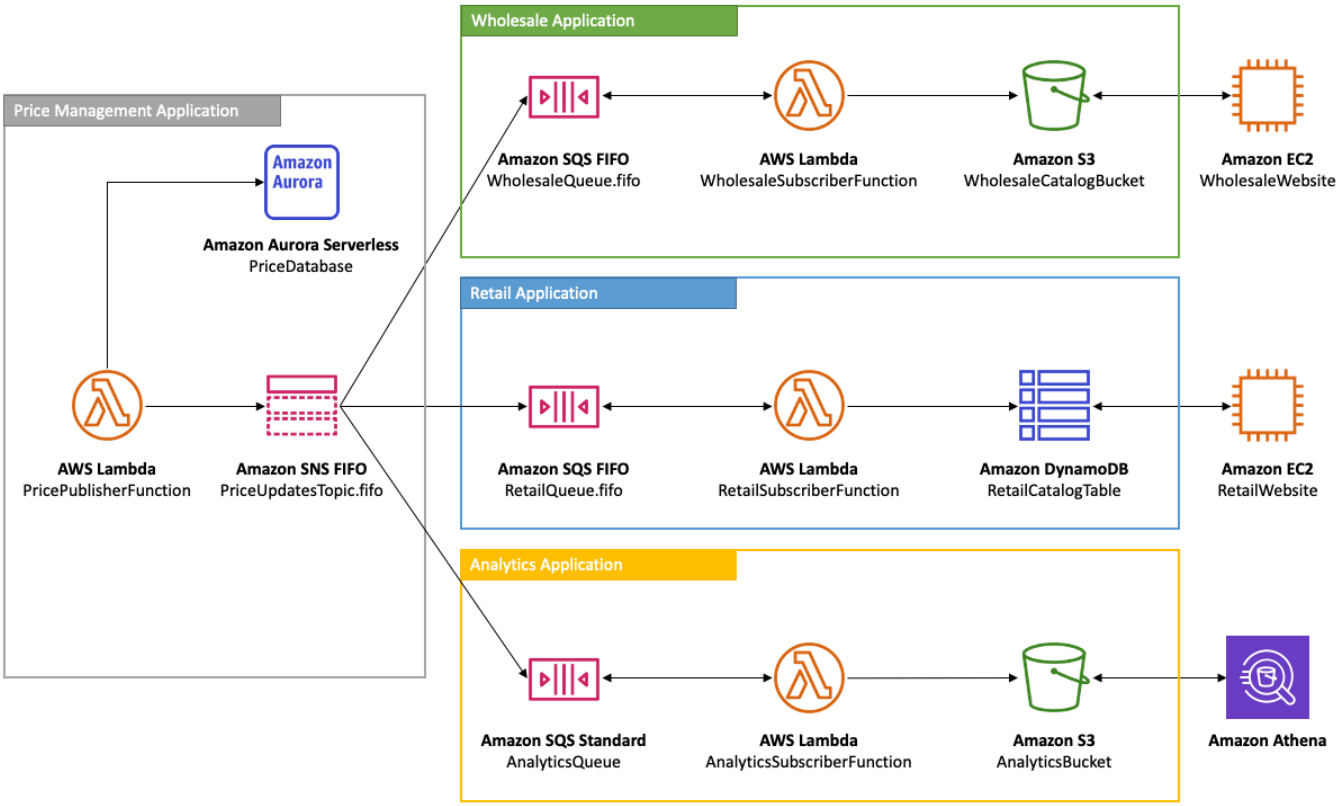
主题

Amazon SNS FIFO 主题示例用例

以下示例描述了汽车零部件制造商使用亚马逊SNSFIFO主题和亚马逊SQS队列构建的电子商务平台。该平台包含四个无服务器应用程序：

- 库存经理使用价格管理应用程序为每件存货设置价格。在该公司，产品价格可能会因汇率波动、市场需求、销售策略的变化而变化。价格管理应用程序使用的 AWS Lambda 功能可在价格发生变化时发布亚马逊SNSFIFO主题的价格更新。

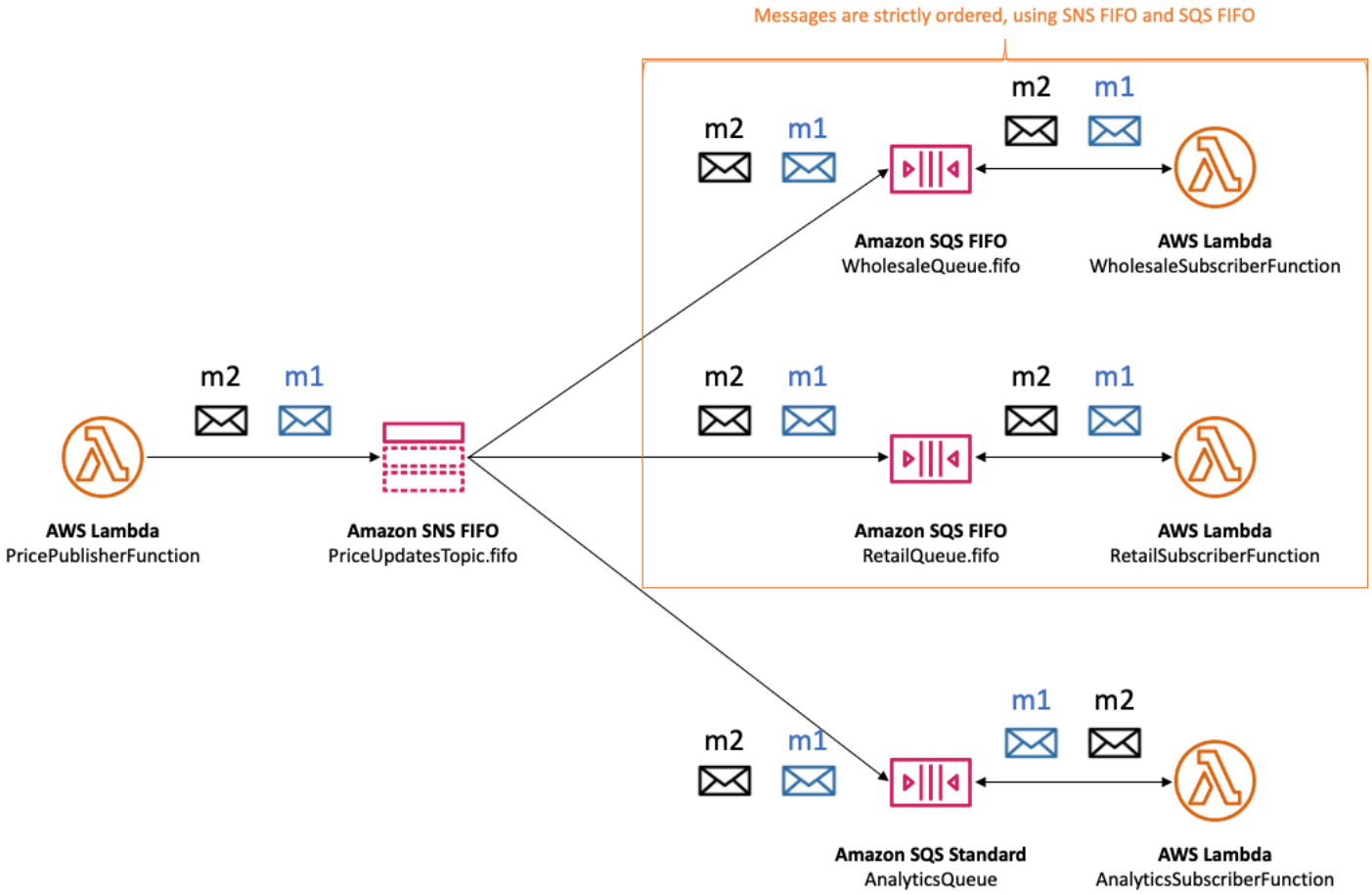
- 批发应用程序为汽车车身修理厂和汽车制造商可以在其中批量购买公司汽车零部件的网站提供后端服务。要获取价格变动通知，批发应用程序将其亚马逊SQSFIFO队列订阅价格管理应用程序的亚马逊SNSFIFO主题。
- 零售应用程序为另一个网站提供后端，车主和汽车改装爱好者可以通过该网站为他们的车辆购买单独的汽车零部件。为了获得价格变动通知，零售应用程序还会将其亚马逊SQSFIFO队列订阅价格管理应用程序的亚马逊SNSFIFO主题。
- 一种分析应用程序，可汇总价格更新并将其存储到 Amazon S3 存储桶中，从而使 Amazon Athena 能够出于商业智能 (BI) 目的查询存储桶。要获取价格变动通知，分析应用程序将其亚马逊SQS标准队列订阅价格管理应用程序的亚马逊SNSFIFO主题。与其他应用程序不同，分析应用程序不需要对价格更新进行严格排序。



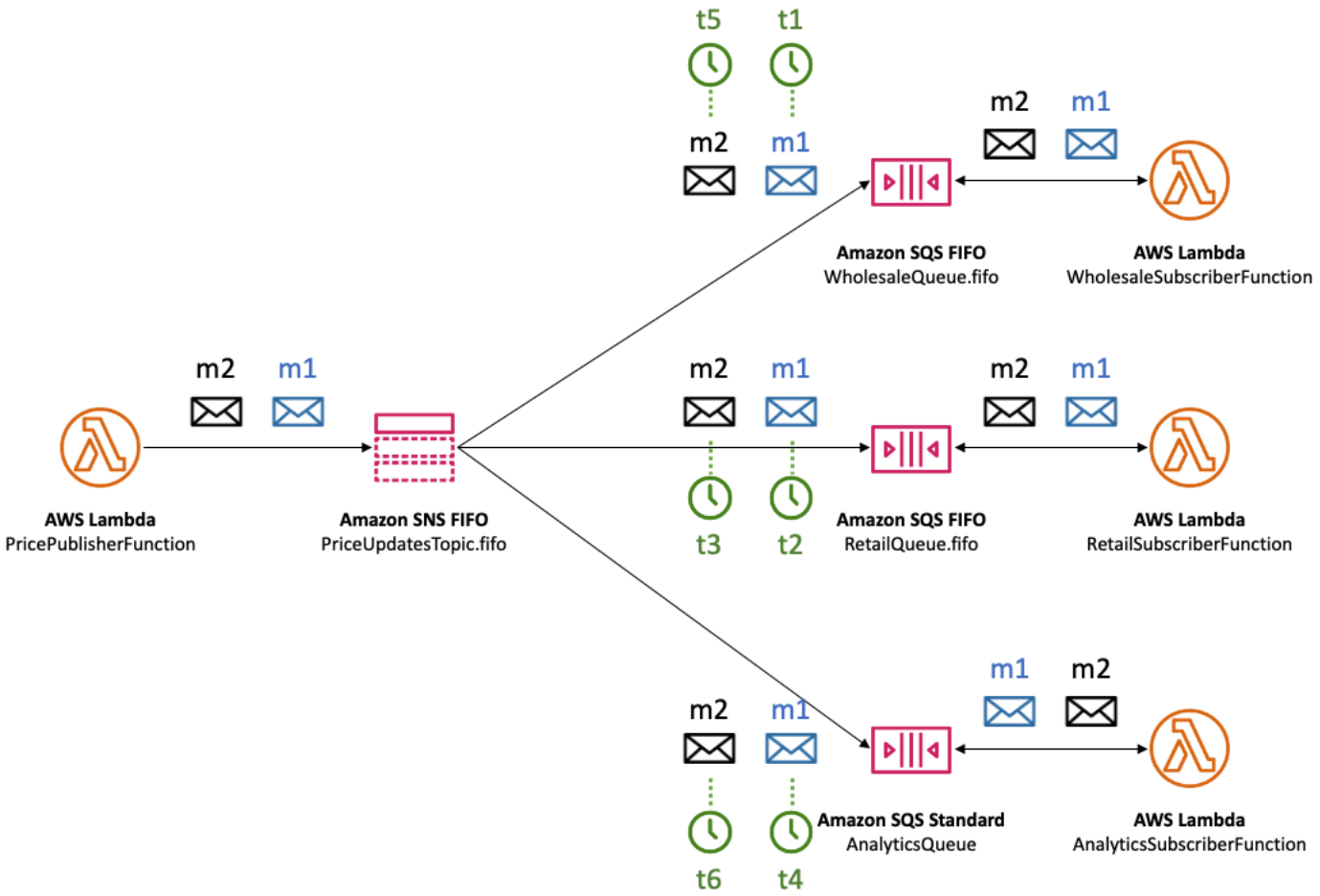
为了使批发和零售应用程序以正确的顺序接收价格更新，价格管理应用程序必须使用严格排序的消息分发系统。使用 Amazon SNS FIFO 主题和 Amazon SQS FIFO 队列可以按顺序处理消息，不会出现重复现象。有关更多信息，请参阅 有关FIFO主题的 Amazon SNS 消息订购详情。有关实现此使用案例的代码片段，请参阅 Amazon FIFO 主题SNS代码示例。

有关FIFO主题的 Amazon SNS 消息订购详情

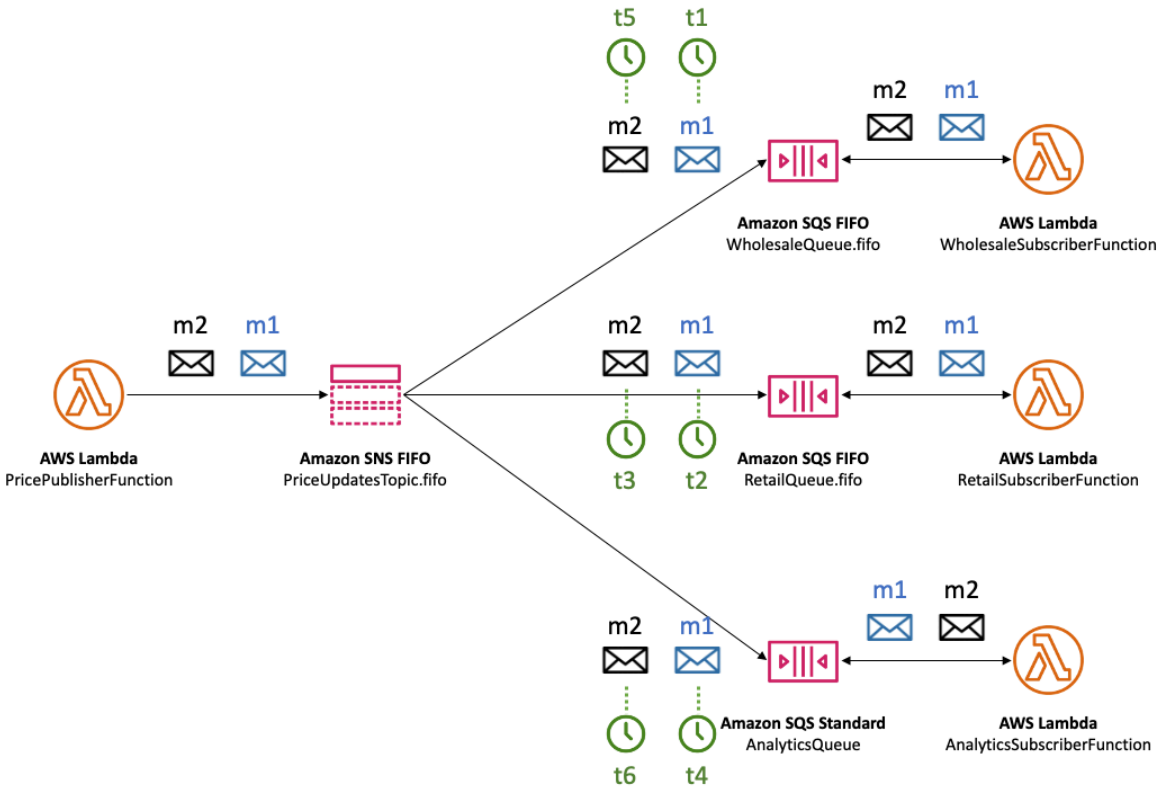
Amazon SNS FIFO 主题始终按照消息发布到该主题的确切顺序向已订阅的亚马逊SQS队列传送消息，而且只能发送一次。订阅了 Amazon SQS FIFO 队列后，队列的使用者将按照消息传送到队列的确切顺序接收消息，并且不会出现重复消息。但是，订阅了 Amazon SQS 标准队列后，该队列的使用者可能会多次收到乱序消息。这可以进一步将订阅者与发布者分开，从而在消息使用和成本优化方面为订阅者提供更大的灵活性，如基于 Amazon SNS FIFO 主题示例用例的下图所示。



请注意，不存在订阅者的隐含排序。以下示例显示消息 m1 首先传输给批发订阅者，然后传输给零售订阅者，再传输给分析订阅者。消息 m2 首先传输给零售订阅者，然后传输给批发订阅者，最后传输给分析订阅者。尽管这两条消息按不同的顺序传送给订阅者，但会保留每个 Amazon 订SQSFIFO阅者的消息顺序。每个订阅者都与任何其他订阅者隔离感知。

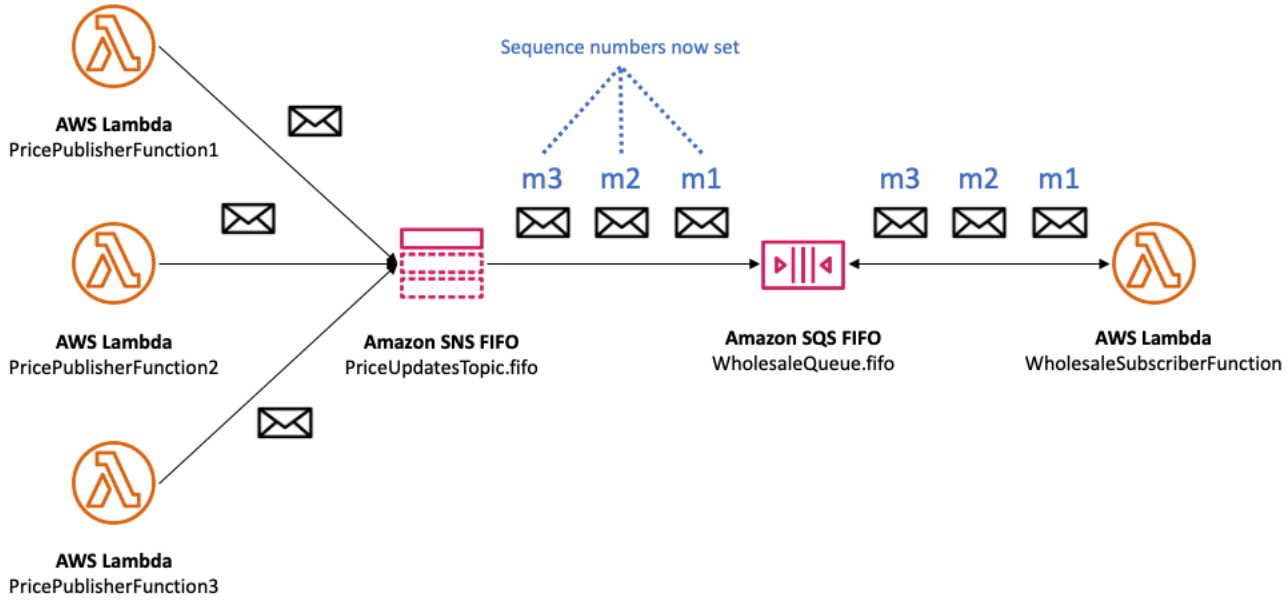


如果 Amazon SQS 队列订阅者无法访问，则可能会失去同步。例如，假设批发应用程序队列所有者错误地更改了亚马逊SQS队列策略，从而阻止了亚马逊SNS服务主体向队列传送消息。在这种情况下，将价格更新传输到批发队列失败，而零售和分析队列的价格更新成功，从而导致订阅者不同步。当批发应用程序队列所有者更正其队列策略时，Amazon SNS 会恢复向订阅队列传送消息。在队列配置不正确时，发布到主题的任何消息都将被删除，除非对应的订阅已配置了死信队列。



您可以让多个应用程序（或同一个应用程序中的多个线程）并行向一个SNSFIFO主题发布消息。当您执行此操作时，实际上是将消息排序委托给 Amazon SNS 服务。要确定已建立的消息序列，您可以检查序列号。

序列号是 Amazon 为每条消息SNS分配的非连续的大型数字。序列号的长度为 128 位，并且每个消息组的序列号会继续增加。序列号作为消息正文的一部分传递给已订阅的 Amazon SQS 队列。但是，如果您启用原始消息传送，则传送到亚马逊SQS队列的消息将不包含序列号或任何其他亚马逊SNS消息元数据。



Amazon SNS FIFO 主题定义了消息组上下文中的排序。有关更多信息，请参阅 针对FIFO主题的 Amazon SNS 消息分组。

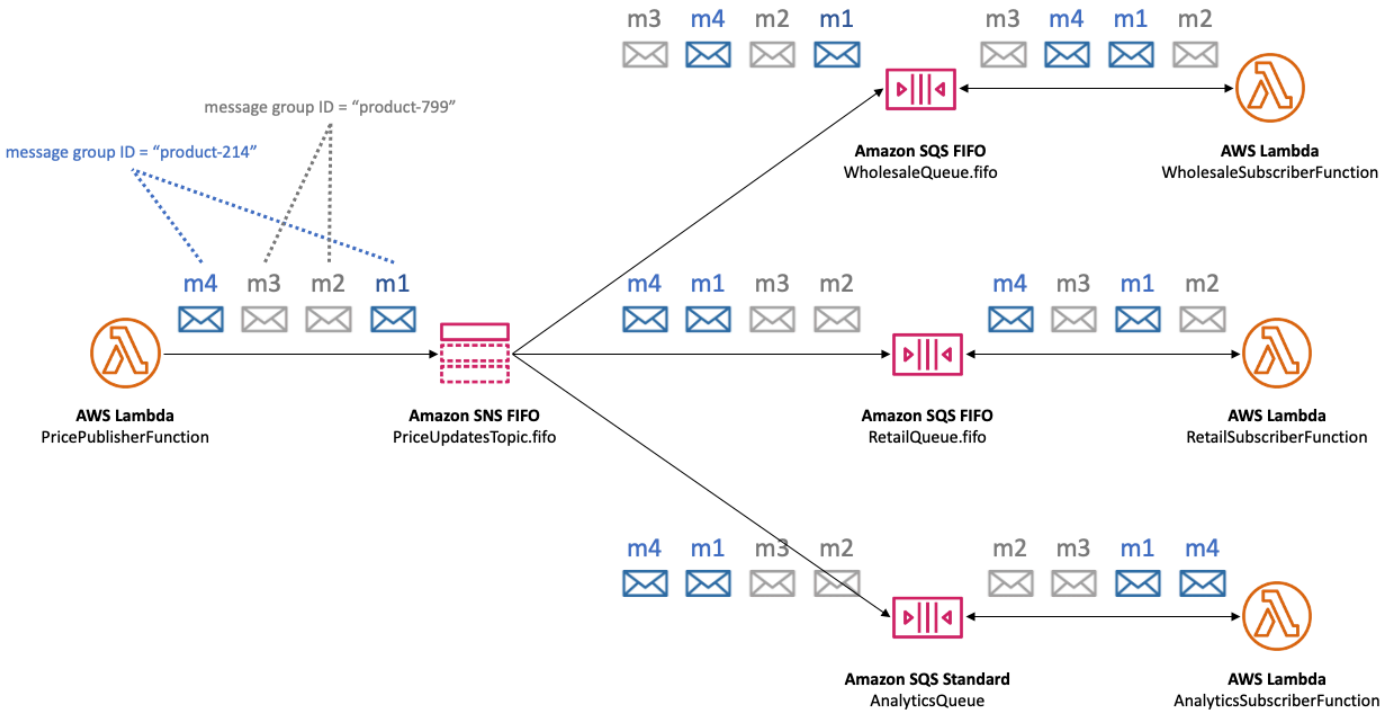
针对FIFO主题的 Amazon SNS 消息分组

属于同一组的消息按相对于组的严格顺序逐个处理。

当您向 Amazon SNS FIFO 主题发布消息时，您需要设置消息组 ID。组 ID 是指定消息属于特定消息组的强制令牌。该SNSFIFO主题将群组 ID 传递给已订阅的 Amazon SQS FIFO 队列。SNSFIFO主题或SQSFIFO队列IDs中的群组数量没有限制。消息组 ID 不会传递给 Amazon SQS 标准队列。

消息组和订阅之间没有关联性。因此，发布到任何消息组的消息都会传输到所有已订阅队列，但须遵守附加到订阅的任何筛选策略。有关更多信息，请参阅 针对FIFO主题的 Amazon SNS 消息传送 和 针对FIFO主题的 Amazon SNS 邮件筛选。

在汽车零部件价格管理示例使用案例中，平台中销售的每个产品都有一个专用的消息组。处理所有价格更新时都使用相同的 Amazon SNS FIFO 主题。价格更新的顺序保留在单个汽车零部件产品的上下文中，但不是跨多个产品。下图演示了工作原理。请注意，对于消息组 ID 为 product-214 的产品，m1 消息始终在 m4 消息之前受到处理。在使用 Amazon 到 Amazon SNS FIFO 的整个工作流程中，都会保留此顺序SQSFIFO。同样，对于消息组 ID 为 produ ct-799 的产品，只要工作流程使用亚马逊和SNSFIFO亚马逊，则消息 m 2 将在消息 m3 之前处理。SQS FIFO但是，使用 Amazon SQS 标准队列时，无法保证消息顺序，也不存在消息组。product-214 和 product-799 消息组彼此独立，因此它们的消息排序方式之间没有任何关系。



按消息组分发数据IDs以提高性能

为了优化传送吞吐量，Amazon SNS FIFO 主题并行传送来自不同消息组的消息，同时严格维护每个消息组内的消息顺序。每个消息组每秒最多可以传送 300 条消息。因此，要实现单个主题的高吞吐量，请使用大量不同的消息组IDs。通过利用各种各样的消息组，Amazon SNS FIFO 主题可以自动将消息分发到更多的并行分区。

> **Note**
>
> Amazon SNS FIFO 主题经过优化，无论消息组的数量如何IDs，均可跨消息组均匀分发消息。AWS 建议您使用大量不同的消息组IDs以优化性能。

当以高吞吐量发布到您的 Amazon SNS FIFO 主题并且订阅了一个或多个 Amazon SQS FIFO 队列时，建议您在队列上启用高吞吐量。有关更多信息，请参阅 Amazon 简单FIFO队列服务开发者指南中的队列高吞吐量。

针对FIFO主题的 Amazon SNS 消息传送

Amazon SNSFIFO（先入先出）主题支持向 Amazon SQS 标准和FIFO队列交付，以便在集成需要近乎实时数据一致性的分布式应用程序时，为客户提供灵活性和控制力。

对于需要保持严格的消息排序或重复数据删除的工作负载，当操作和事件的顺序至关重要或不能容忍重复时，Amazon SNS FIFO 主题与订阅为传输终端节点的 Amazon SQS FIFO 队列相结合，可以增强应用程序之间的消息传送能力。

对于允许尽最大努力订购和 at-least-once 交付的工作负载，订阅 A SQS mazon 标准队列订阅 SNS FIFO Amazon 主题可以降低成本，此外还可以在未使用的工作负载之间共享队列。FIFO

> **Note**
>
> 要将消息从 Amazon SNS FIFO 主题分散到 AWS Lambda 函数，需要采取额外的步骤。首先，向 Amazon SQS FIFO 或标准队列订阅该主题。然后，配置队列以触发函数。有关更多信息，请参阅 C AWS ompute 博客上的 “SQSFIFO作为事件源” 一文。

SNSFIFO主题无法将消息传送到客户管理的端点，例如电子邮件地址、移动应用程序、短信的电话号码 (SMS) 或 HTTP (S) 端点。这些终端节点类型不能保证保留严格的消息排序。尝试为客户管理的终端节点订阅SNSFIFO主题会导致错误。

SNSFIFO主题支持与标准主题相同的邮件筛选功能。有关更多信息，请参阅 针对FIFO主题的 Amazon SNS 邮件筛选 Comp AWS ut e 博客上的 “使用 Amazon 消息筛选简化您的 Pub/Sub SNS 消息” 一文。

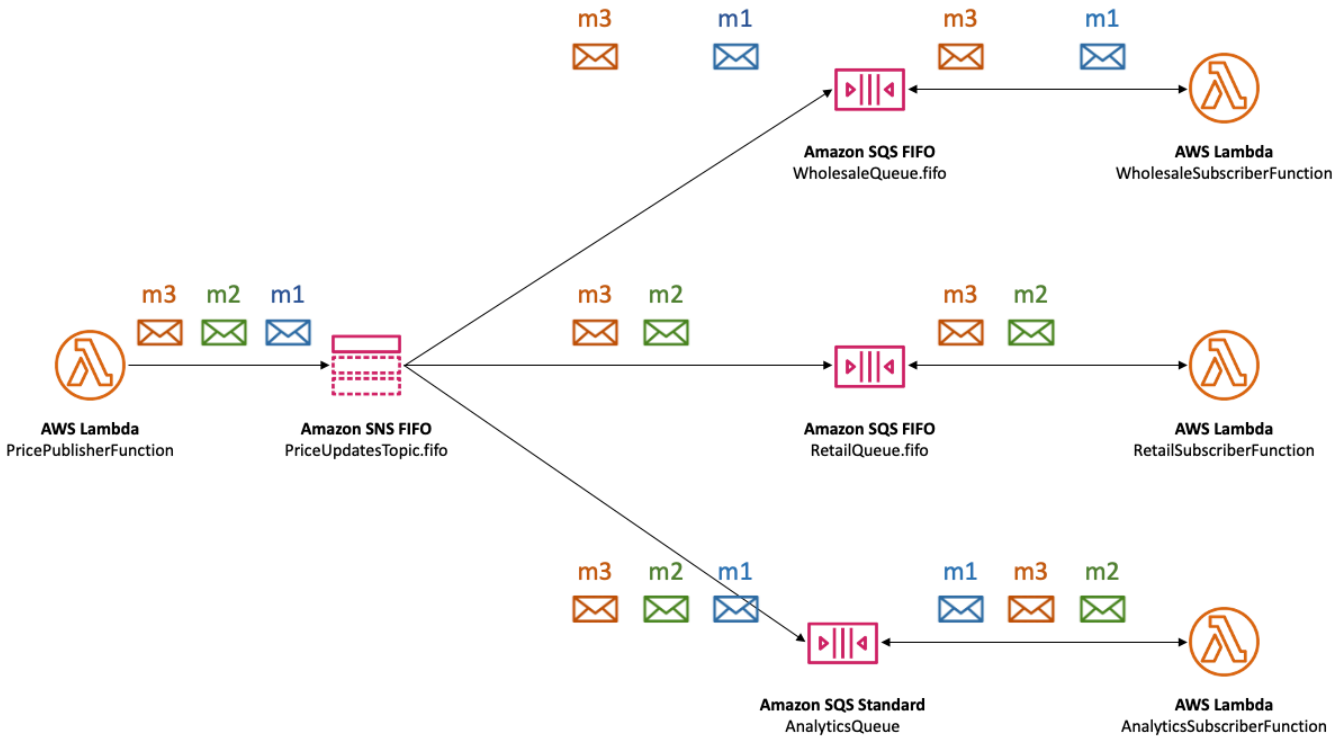
针对FIFO主题的 Amazon SNS 邮件筛选

Amazon SNS FIFO 主题支持消息筛选。使用消息筛选可通过从发布者系统卸载消息路由逻辑，从订阅者系统卸载消息筛选逻辑来简化您的架构。

当您为 Amazon SQS FIFO 或标准队列订阅某个SNSFIFO主题时，您可以使用消息筛选来指定订阅者接收的消息子集，而不是全部消息。每个订阅者都可以将其自己的筛选策略设置为订阅属性。根据筛选策略的范围，筛选策略与入站消息属性或消息正文进行匹配。如果筛选策略匹配，则主题会向订阅者传输消息的副本。如果没有匹配项，则主题不会传输消息的副本。

在汽车零件价格管理示例用例中，假设设置了以下亚马逊SNS筛选政策，并且筛选策略的范围为`MessageBody`：

- 对于批发队列，筛选策略 {"business":["wholesale"]} 匹配包含名为 business 的键且在一组值中具有 wholesale 的每条消息。在下图中，消息 m1 中的键之一是值为 wholesale 的 business。消息 m3 中的键之一是值为 ["wholesale,retail"] 的 business。因此，m1 和 m3 均匹配筛选策略的条件，并且这两条消息都会传输到批发队列中。
- 对于零售队列，筛选策略 {"business":["retail"]} 匹配包含名为 business 的键且在一组值中具有 retail 的每条消息。在图中，消息 m2 中的键之一是值为 retail 的 business。消息 m3 中的键之一是值为 ["wholesale,retail"] 的 business。因此，m2 和 m3 均匹配筛选策略的条件，并且这两条消息都会传输到零售队列中。
- 对于分析队列，我们希望 Amazon Athena 接收所有记录，因此不应用任何筛选策略。



SNSFIFO主题支持各种匹配运算符，包括属性字符串值、属性数值和属性键。有关更多信息，请参阅 Amazon SNS 邮件过滤。

SNSFIFO主题不会向订阅的终端节点传送重复的消息。有关更多信息，请参阅 针对FIFO主题的 Amazon SNS 消息重复数据删除。

针对FIFO主题的 Amazon SNS 消息重复数据删除

Amazon SNS FIFO 主题和 Amazon SQS FIFO 队列支持消息重复数据删除，只要满足以下条件，即可提供精确一次的消息传送和处理：

- 已订阅的 Amazon SQS FIFO 队列存在且拥有允许亚马逊SNS服务主体向该队列传送消息的权限。
- Amazon SQS FIFO 队列使用者处理消息并在可见性超时到期之前将其从队列中删除。
- Amazon SNS 订阅主题没有消息筛选功能。当您配置邮件筛选时，Amazon SNS FIFO 主题支持传 at-most-once 送，因为可以根据您的订阅筛选策略筛选出消息。
- 没有阻止确认邮件传输的网络中断。

> **Note**
>
> 消息重复数据删除适用于整个 Amazon SNS FIFO 主题，而不适用于单个消息组。

当您向 Amazon SNS FIFO 主题发布消息时，该消息必须包含重复数据删除 ID。此 ID 包含在亚马逊SNSFIFO主题发送给已订阅的亚马逊SQSFIFO队列的消息中。

如果成功将具有特定重复数据删除 ID 的消息发布到 Amazon SNS FIFO 主题，则在五分钟重复数据删除间隔内，使用相同重复数据删除 ID 发布的任何消息都将被接受但不会传送。即使消息已传送到订阅的终端节点，Amazon SNS FIFO 主题仍会继续跟踪消息重复数据删除 ID。

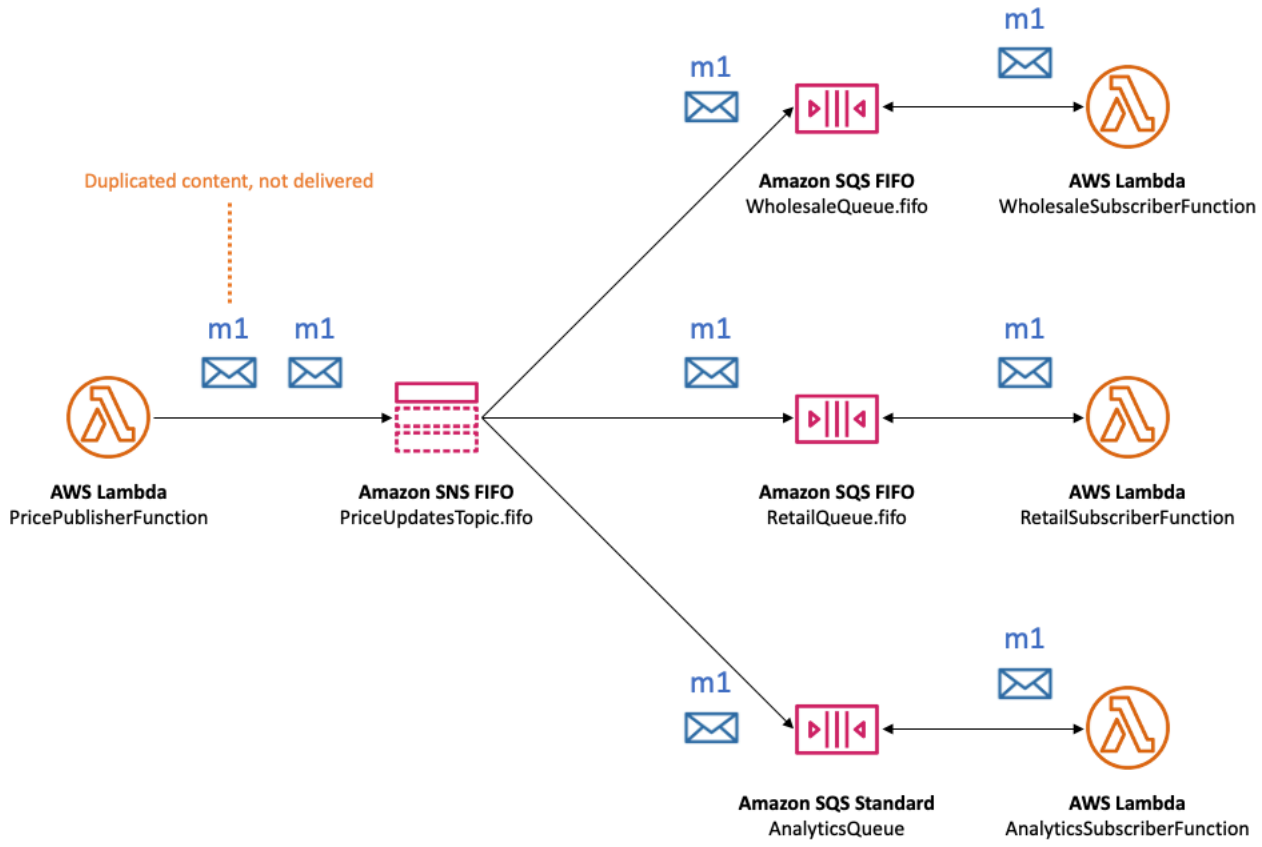
如果保证每条已发布消息的消息正文都是唯一的，则可以为亚马逊SNSFIFO主题和订阅的亚马逊队列启用基于内容的重复数据删除。SQS FIFOAmazon SNS 使用消息正文生成一个唯一的哈希值，用作每条消息的重复数据删除 ID，因此您无需在发送每条消息时设置重复数据删除 ID。

> **Note**
>
> 消息属性不包括在哈希计算中。

如果为 Amazon SNS FIFO 主题启用基于内容的重复数据删除，并且发布了带有重复数据删除 ID 的消息，则发布的重复数据删除 ID 将覆盖生成的基于内容的重复数据删除 ID。

在汽车零部件价格管理示例使用案例中，公司必须为每次价格更新设置一个通用唯一的重复数据删除 ID。这是因为即使在批发和零售的消息属性不同时，消息正文也可以是相同的。但是，如果公司将业务

类型（批发或零售）与商品编码和产品价格一起添加到消息正文中，则他们可以在亚马逊SNSFIFO主题和订阅的亚马逊队列中启用基于内容的复制。SQS FIFO



除了消息排序和重复数据删除外，Amazon SNS FIFO 主题还支持使用 AWS KMS 密钥进行消息服务器端加密 (SSE)，以及通过VPC终端节点进行消息隐私。 AWS PrivateLink有关更多信息，请参阅 针对FIFO主题的 Amazon SNS 消息安全。

针对FIFO主题的 Amazon SNS 消息安全

您可以选择让 Amazon SNS 和 Amazon 使用 AWS Key Management Service (AWS KMS) 客户主密钥 (CMKs) 对发送到FIFO主题和队列的消息进行SQS加密。您可以创建加密的FIFO主题和队列，也可以选择加密现有FIFO主题和队列。Amazon SNS 和 Amazon 仅SQS对邮件正文进行加密。它们不加密消息属性、资源元数据或资源指标。

> **Note**
>
> 向现有FIFO主题或队列添加加密不会加密任何积压的消息，而从主题或队列中移除加密会使积压的消息处于加密状态。

SNSFIFO主题会在将消息传送到订阅的终端节点之前立即对其进行解密。SQSFIFO队列在将消息返回给使用者应用程序之前对其进行解密。有关更多信息，请参阅 C AWS ompute Bl og 上发布到 Amazon SNS 的加密消息亚马逊SNS数据加密以及 AWS KMS帖子。

此外，SNSFIFO主题和SQSFIFO队列支持消息隐私，接口VPC端点由提供支持 AWS PrivateLink。使用接口终端节点，您可以将消息从 Amazon Virtual Private Cloud (AmazonVPC) 子网发送到FIFO主题和队列，而无需通过公共互联网。这种模式将您的消息传递保存在 AWS 基础设施和网络中，从而增强了应用程序的整体安全性。使用时 AWS PrivateLink，您无需设置互联网网关、网络地址转换 (NAT) 或虚拟专用网络 (VPN)。有关更多信息，请参阅使用VPC终端节点保护 Amazon SNS 流量AWS 安全博客上发布的SNS带有 AWS PrivateLink帖子的 Amazon 安全消息。

SNSFIFO主题还支持跨可用区的死信队列和消息存储。有关更多信息，请参阅 FIFO主题的 Amazon SNS 消息持久性。

FIFO主题的 Amazon SNS 消息持久性

亚马逊SNSFIFO主题和亚马逊SQS队列是持久的。这两种资源类型都以冗余方式跨多个可用区存储消息，并提供死信队列以处理异常情况。

在亚马逊中SNS，当亚马逊SNS主题由于客户端或服务器端错误而无法访问已订阅的亚马逊SQS队列时，消息传递就会失败：

- 当 Amazon SNS FIFO 主题包含过时的订阅元数据时，就会发生客户端错误。导致客户端错误的两个常见原因是 Amazon SQS 队列所有者执行以下操作之一：
  - 删除队列。
  - 更改队列策略时会阻止 Amazon SNS 服务主体向其发送消息。

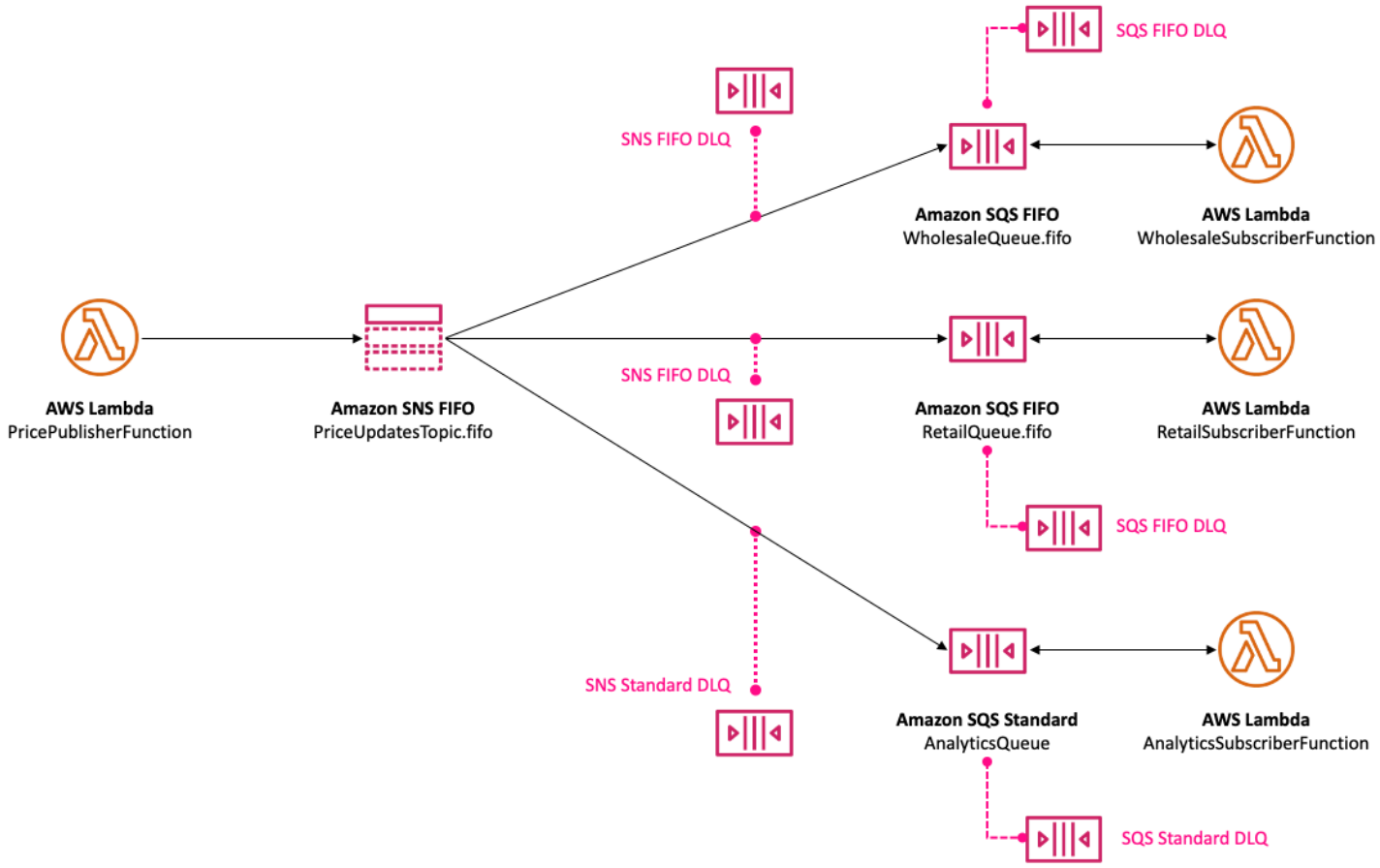
  Amazon SNS 不会重试发送由于客户端错误而失败的消息。
- 在以下情况下可能会发生服务器端错误：
  - Amazon SQS 服务不可用。
  - 亚马逊SQS无法处理来自亚马逊SNS服务的有效请求。

  当服务器端出现错误时，Amazon SNS FIFO 主题会在 23 天内重试失败的配送次数 100,015 次。有关更多信息，请参阅 Amazon SNS 消息传送重试次数。

对于任何类型的错误，亚马逊SNS都可以将消息留给亚马逊SQS死信队列，这样数据就不会丢失。

在 Amazon 中SQS，当使用者应用程序无法接收、处理消息并将其从队列中删除时，消息处理就会失败。当最大数量的接收请求失败时，Amazon SQS 可以将消息留给死信队列，这样数据就不会丢失。

在汽车零部件价格管理示例用例中，公司可以为每个亚马逊SNSFIFO主题订阅以及每个订阅的亚马逊队列分配一个亚马逊SQSSQS死信队列 (DLQ)。这可以保护公司免受任何价格更新损失。



与亚马逊SNS订阅关联的死信队列必须是与订阅SQS队列相同类型的亚马逊队列。例如，亚马逊SNSFIFO订阅的亚马逊SQSFIFO队列必须将亚马逊SQSFIFO队列作为死信队列。同样，亚马逊SNSFIFO订阅的亚马逊SQS标准队列必须使用亚马逊SQS标准队列作为其死信队列。有关更多信息，请参阅 Amazon SNS 死信队列 Comp AWS ute 博客上的 “DLQs为亚马逊设计耐用的无服务器应用程序” SNSSQS， AWS Lambda Amazon。

为了延长持久性以帮助从下游故障中恢复，主题所有者还可以使用FIFO主题将消息存档长达 365 天。然后，主题订阅用户可以将这些消息重播到已订阅的端点，以恢复因下游应用程序故障而丢失的消息，或者复制现有应用程序的状态。有关更多信息，请参阅Amazon SNS 消息存档和重播FIFO主题。

Amazon SNS 消息存档和重播FIFO主题

主题

什么是消息归档与重播功能？

Amazon SNS 消息存档和重播是一种无需编码、就地的消息存档，允许主题所有者在其主题中存储（或存档）消息。然后，主题订阅用户可以将归档的消息检索（或重播）回订阅的端点，这可用于：

- 恢复可能由于下游应用程序故障而丢失的消息。
- 通过订阅新端点并选择要从中复制的所需时间戳，将现有应用程序的状态复制到新应用程序。

您可以将邮件存档和重播与 AWS API、SDK AWS CloudFormation、和一起使用 AWS Management Console。

> **Note**
>
> Amazon SNS 消息存档和重播仅适用于 application-to-application (A2A) FIFO 主题。

消息归档与重播功能由两个主要部分组成：

1. 消息归档 - 主题所有者对主题启用归档与重播特征，并设置消息保留期（最长 365 天）。主题所有者还可以使用 Amazon CloudWatch 指标监控归档的消息。有关更多信息，请参阅为FIFO主题所有者提供的 Amazon SNS 消息存档。
2. 消息重播 - 主题订阅用户启动一组消息从主题到其订阅端点的重播。有关更多信息，请参阅为FIFO主题订阅者提供的 Amazon SNS 消息重播。

为FIFO主题所有者提供的 Amazon SNS 消息存档

通过消息归档，您可以归档发布到您的主题的所有消息的单个副本。您可以通过对主题启用消息归档策略将已发布的消息存储在主题中，该策略将为链接到该主题的所有订阅启用消息归档。消息可以归档至少一天，最多 365 天。

设置归档策略时需支付额外费用。有关定价信息，请参阅 Amazon SNS 定价。

主题

- 使用创建邮件存档策略 AWS Management Console
- 使用创建邮件存档策略 API
- 使用创建邮件存档策略 SDK
- 使用创建邮件存档策略 AWS CloudFormation
- 授予对加密归档的访问权限
- 使用 Amazon 监控邮件存档指标 CloudWatch

使用创建邮件存档策略 AWS Management Console

使用此选项，通过 AWS Management Console创建新的消息归档策略。

1. 登录 Amazon SNS 控制台。
2. 选择一个主题或创建一个新主题。要了解有关创建主题的更多信息，请参阅创建 Amazon SNS 主题。

   > **Note**
   >
   > Amazon SNS 消息存档和重播仅适用于 application-to-application (A2A) FIFO 主题。

3. 在编辑主题页面上，展开归档策略部分。
4. 启用归档策略特征，然后输入要在主题中归档消息的天数。
5. 选择 Save changes（保存更改）。

查看、编辑和停用消息归档主题策略

- 在主题详细信息页面上，保留策略显示归档策略的状态，包括设置该策略对应的天数。选择归档策略选项卡以查看以下消息归档详细信息：
  - 状态 - 应用归档策略后，归档与重播功能状态显示为活动。当存档策略设置为空JSON对象时，存档和重播状态显示为非活动状态。
  - 消息保留期 - 指定的消息保留天数。
  - 归档开始日期 - 订阅用户可以重播消息的起始日期。
  - JSON预览-存档策略的JSON预览。

- （可选）要编辑归档策略，请转到主题摘要页面并选择编辑。
- （可选）要停用归档策略，请转至主题摘要页面并选择编辑。停用归档策略并选择保存更改。
- （可选）要使用归档策略删除主题，必须先按照前面所述停用归档策略。

> **Important**
>
> 为避免意外删除消息，您不能在具有有效消息归档策略的情况下删除主题。必须先停用主题的消息归档策略，然后才能删除该主题。当您停用邮件存档策略时，Amazon SNS 会删除所有已存档的邮件。删除主题时，订阅将被删除，并且任何传输中的消息都可能无法传送。

使用创建邮件存档策略 API

要使用创建消息存档策略API，您需要将该属性`ArchivePolicy`添加到您的主题中。您可以使用API操作`CreateTopic`和`ArchivePolicy`进行设置`SetTopicAttributes`。 `ArchivePolicy`只有一个值`MessageRetentionPeriod`，它表示 Amazon SNS 保留消息的天数。要为主题激活消息归档，请将 `MessageRetentionPeriod` 设置为大于零的整数值。例如，要将归档中的消息保留 30 天，请将 `ArchivePolicy` 设置为：

```
{
    "ArchivePolicy": {
        "MessageRetentionPeriod": "30"
    }
}
```

要对主题禁用消息归档并清除归档，请取消设置 `ArchivePolicy`，如下所示：

```
{}
```

使用创建邮件存档策略 SDK

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》config和《工具参考指南》中的共享 AWS SDKs和*credentials*文件。

以下代码示例说明如何将 Amazon SNS 主题的设置`ArchivePolicy`为将发布到该主题的所有消息保留 30 天。

```
// Specify the ARN of the Amazon SNS topic to set the ArchivePolicy for.
```

```
String topicArn =
    "arn:aws:sns:us-east-2:123456789012:MyArchiveTopic.fifo";

// Set the MessageRetentionPeriod to 30 days for the ArchivePolicy.
String archivePolicy =
    "{\"MessageRetentionPeriod\":\"30\"}";

// Set the ArchivePolicy for the Amazon SNS topic
SetTopicAttributesRequest request = new SetTopicAttributesRequest()
    .withTopicArn(topicArn)
    .withAttributeName("ArchivePolicy")
    .withAttributeValue(archivePolicy);
sns.setTopicAttributes(request);
```

使用创建邮件存档策略 AWS CloudFormation

要使用创建存档策略， AWS CloudFormation 请参阅AWS CloudFormation 用户指南*AWS::SNS::Topic*中的。

授予对加密归档的访问权限

必须先完成以下步骤，然后订阅用户才能开始重播来自加密主题的消息。由于过去的消息会被重播，因此SNS需要向 Amazon 提供`Decrypt`访问用于加密存档中消息的密KMS钥的权限。

1. 当您使用KMS密钥加密消息并将其存储在主题中时，必须通过密钥策略授予 Amazon SNS 解密这些消息的权限。有关更多信息，请参阅向 Amazon 授予解密权限 SNS。
2. AWS KMS 为亚马逊启用SNS。有关更多信息，请参阅配置 AWS KMS 权限。

> **Important**
>
> 在向KMS密钥策略中添加新部分时，请勿更改策略中的任何现有部分。如果对某个主题启用了加密，KMS密钥被禁用或删除，或者没有为亚马逊正确配置KMS密钥策略SNS，Amazon 将SNS无法向您的订阅者重播消息。

向 Amazon 授予解密权限 SNS

SNS要让 Amazon 从您的主题存档访问加密消息并将其重播到已订阅的终端节点，您必须启用 Amazon SNS 服务原则来解密这些消息。

以下是允许Amazon SNS 服务主体在重播您的主题内的历史消息时解密存储的消息所需的示例策略。

```
{
    "Sid": "Allow SNS to decrypt archived messages",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

使用 Amazon 监控邮件存档指标 CloudWatch

您可以使用以下指标在 Amaz CloudWatch on 上监控存档的邮件。为了收到工作负载异常的通知并帮助避免影响，您可以根据这些指标配置 Amazon CloudWatch 警报。有关更多详细信息，请参阅在 Amazon 中记录和监控 SNS。

指标	描述
ApproximateNumberOfMessagesArchived	以 60 分钟的分辨率向主题所有者提供在主题归档中归档的消息总数。
ApproximateNumberOfBytesArchived	以 60 分钟的分辨率向主题所有者提供对主题归档中的所有消息归档的总字节数。
NumberOfMessagesArchiveProcessing	以 1 分钟的分辨率向主题所有者提供在间隔期间保存到主题存档的消息数。
NumberOfBytesArchiveProcessing	以 1 分钟的分辨率向主题所有者提供在间隔期间保存到主题存档的总字节数。

`GetTopicAttributes`API有一个`BeginningArchiveTime`属性，它表示订阅者可以开始重播的最早时间戳。以下是此API操作的示例响应：

```
{
```

```
  "ArchivePolicy": {
     "MessageRetentionPeriod": "<integer>"
   },
   "BeginningArchiveTime": "<timestamp>",
   ...
}
```

为FIFO主题订阅者提供的 Amazon SNS 消息重播

Amazon re SNS play 允许主题订阅者从主题数据存储中检索存档的消息，然后将其重新传送（或重播）到已订阅的终端节点。创建订阅后即可立即重播消息。重播的消息与原始副本具有相同的内容、MessageId 和 Timestamp，还包含属性 Replayed，以帮助您识别这是一条重播的消息。要仅重播精选消息，可以在订阅中添加筛选策略。有关筛选消息的更多信息，请参阅筛选重播的消息。

主题

- 使用创建消息重播策略 AWS Management Console
- 使用向订阅添加重播政策 API
- 使用向订阅添加重播政策 SDK
- 理解 EndingPoint
- 筛选重播的消息
- 使用 Amazon 监控消息重播指标 CloudWatch

使用创建消息重播策略 AWS Management Console

使用此选项，通过 AWS Management Console创建新的重播策略。

1. 登录 Amazon SNS 控制台。
2. 选择一个主题订阅或创建一个新的主题订阅。要了解有关创建订阅的更多信息，请参阅创建对 Amazon SNS 主题的订阅。
3. 要启动消息重播，请转到重播下拉列表并选择开始重播。
4. 从重播时间范围模式中，进行以下选择：

   a. 选择重播开始日期和时间-选择要开始重播存档邮件的日期（YYYY/MM/DD 格式）和时间（24 小时 hh: mm: ss 格式）。开始时间应晚于近似归档时间的开始时间。

   b. （可选）选择重播结束日期和时间-选择要停止重播存档邮件的日期（YYYY/MM/DD 格式）和时间（24 小时 hh: mm: ss 格式）。

c. 选择开始重播。

5. （可选）要停止消息重播，请转到订阅详细信息页面，然后从重播下拉列表中选择停止重播。
6. （可选）要使用监控此工作流程中的邮件重播指标 CloudWatch，请参阅使用 Amazon 监控消息重播指标 CloudWatch。

查看和编辑消息重播策略

您可以从订阅详细信息页面执行以下操作：

- 要查看消息重播状态，重播状态字段将显示以下值：
  - 已完成 - 重播已成功重新传送所有消息，现在正在传送新发布的消息。
  - 进行中 – 重播当前正在重播所选消息。
  - 失败 – 重播无法完成。
  - 待处理 - 重播启动时的默认状态。
- （可选）要修改消息重播策略，请转到订阅详细信息页面，然后从重播下拉列表中选择开始重播。开始某个重播将取代现有的重播。

使用向订阅添加重播政策 API

要重播存档的消息，请使用属性ReplayPolicy。 ReplayPolicy可以与Subscribe和SetSubscriptionAttributesAPI操作一起使用。此策略包含以下值：

- StartingPoint（必需）- 表示从何处开始重播消息。
- EndingPoint（可选）- 表示何时停止重播消息。如果省略 EndingPoint，则重播将继续，直到赶上当前时间。
- PointType（必需）- 设置起点和终点的类型。目前，PointType 唯一支持的值是 Timestamp。

例如，要从 2023 年 10 月 1 日下游故障中恢复并重新发送两小时内的所有消息，请使用SetSubscriptionAttributesAPI操作按ReplayPolicy如下方式进行设置：

```
{
  "PointType":"Timestamp",
  "StartingPoint":"2023-10-01T10:00:00.000Z",
  "EndingPoint":"2023-10-01T12:00:00.000Z"
}
```

要重播截至 2023 年 10 月 1 日发送到该主题的所有消息，并继续接收与您的主题有关的所有新发布的消息，请使用SetSubscriptionAttributesAPI操作ReplayPolicy对您的订阅进行设置，如下所示：

```
{
  "PointType":"Timestamp",
  "StartingPoint":"2023-10-01T00:00:00.000Z"
}
```

为了验证消息是否已重播，将在每条重播的消息中添加布尔属性 Replayed。

使用向订阅添加重播政策 SDK

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》config和《工具参考指南》中的共享 AWS SDKs和*credentials*文件。

以下代码示例显示了如何将订阅设置为ReplayPolicy在 2023 年 10 月 1 日的 2 小时内重新传送来自亚马逊SNSFIFO主题存档的消息。

```
// Specify the ARN of the Amazon SNS subscription to initiate the ReplayPolicy on.
String subscriptionArn =
    "arn:aws:sns:us-
east-2:123456789012:MyArchiveTopic.fifo:1d2a3e9d-7f2f-447c-88ae-03f1c68294da";

// Set the ReplayPolicy to replay messages from the topic's archive
// for a 2 hour time period on October 1st 2023 between 10am and 12pm UTC.
String replayPolicy =
    "{\"PointType\":\"Timestamp\",\"StartingPoint\":\"2023-10-01T10:00:00.000Z\",
\"EndingPoint\":\"2023-10-01T12:00:00.000Z\"}";

// Set the ArchivePolicy for the Amazon SNS topic
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("ReplayPolicy")
    .withAttributeValue(replayPolicy);
sns.setSubscriptionAttributes(request);
```

理解 EndingPoint

当您向 Amazon SNS 订ReplayPolicy阅应用时，该EndingPoint值是可选的。如果未EndingPoint提供，则重播将从指定的时间开始StartingPoint并持续到当前时间，包括处理任何新发布的消息。一旦赶上，订阅将作为常规订阅运行，在新消息发布时接收新消息。

如果指定EndingPoint了，则该服务将从StartingPoint上到向重播消息，EndingPoint然后停止。此操作实际上会暂停订阅。订阅暂停期间，新发布的消息将不会传送到已订阅的终端节点。

要恢复邮件传送，请在ReplayPolicy不提供消息的情况下应用新的EndingPointStartingPoint，并将设置为继续接收消息的所需时间点。例如，要从之前的重播结束的地方恢复订阅，StartingPoint请将新的设置为先前提供的订阅EndingPoint。

筛选重播的消息

Amazon SNS 消息筛选允许您控制亚马逊向您的订阅者终端SNS节点重播的消息。当消息筛选和消息存档都启用时，Amazon 会SNS首先从主题的数据存储中检索消息，然后将消息应用于订阅的数据存储中。FilterPolicy当存在匹配项时，消息将传送到订阅的端点，否则消息将被筛选掉。有关更多信息，请参阅 亚马逊SNS订阅筛选政策。

使用 Amazon 监控消息重播指标 CloudWatch

您可以使用以下指标在 Amaz CloudWatch on 上监控重播消息。为了收到工作负载异常的通知并帮助避免影响，您可以根据这些指标配置 Amazon CloudWatch 警报。有关更多详细信息，请参阅在 Amazon 中记录和监控 SNS。

指标	描述
NumberOfReplayedNotificationsDelivered	以 1 分钟的分辨率向订阅用户提供主题归档中重播的消息总数。
NumberOfReplayedNotificationsFailed	以 1 分钟的分辨率向订阅用户提供主题归档中未能传送的已重播消息的总数。

Amazon FIFO 主题SNS代码示例

您可以使用以下代码示例将使用亚马逊SNSFIFO主题的汽车零件价格管理示例用例与亚马逊SQSFIFO队列或标准队列进行集成。

主题

- 使用 AWS SDK
- 使用 AWS CloudFormation

使用 AWS SDK

使用时 AWS SDK，您可以通过将其`FifoTopic`属性设置为来创建亚马逊SNSFIFO主题**`true`**。您可以通过将 Amazon SQS FIFO 队列的`FifoQueue`属性设置为来创建该队列**`true`**。此外，您还必须在每个FIFO资源的名称中添加**`.fifo`**后缀。创建FIFO主题或队列后，您无法将其转换为标准主题或队列。

以下代码示例创建了这些资源FIFO和标准队列资源：

- 发布价格SNSFIFO更新的 Amazon 主题
- 为批发和零售应用程序提供这些更新的Amazon SQS FIFO 队列
- 用于存储记录的分析应用程序的 Amazon SQS 标准队列，可以查询这些记录以获取商业智能 (BI)
- 将三个队列与主题关联的 Amazon SNS FIFO 订阅

本示例将设置订阅中的筛选条件策略。如果通过向主题发布消息来测试示例，请确保您发布的是带 `business` 属性的消息。为属性值指定 `retail` 或 `wholesale`。否则，消息将被筛选掉，且不会传递到订阅的队列中。有关更多信息，请参阅 针对FIFO主题的 Amazon SNS 邮件筛选。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

此示例

- 创建一个 Amazon SNS FIFO 主题、两个亚马逊SQSFIFO队列和一个标准队列。
- 将队列订阅到主题，发布一条消息到主题。

该测试验证每个队列是否收到消息。完整的示例还显示了添加访问策略，并在最后删除了资源。

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {
```

```
        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);
```

```
        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                    .topicArn(topicARN)
                    .endpoint(queue.queueARN)
                    .protocol("sqs")
                    .build();

            // Subscribe to the endpoint by using the SNS service client.
```

```
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(attributeValue)
                    .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                    .topicArn(topicArn)
                    .subject(subject)
                    .message(payload)
                    .messageGroupId(groupId)
                    .messageDeduplicationId(dedupId)
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- 有关API详细信息，请参阅 “参AWS SDK for Java 2.x API考” 中的以下主题。
  - CreateTopic
  - Publish
  - 订阅

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建亚马逊SNSFIFO主题，为该主题订阅亚马逊SQSFIFO和标准队列，然后向该主题发布消息。

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
```

```
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")
```

```
    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
 letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
 characters long.
        For a FIFO topic, the name must end with the .fifo suffix.
```

```
        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
```

```
                            }
                        ],
                    }
                )
            }
        )
        logger.info("Added trust policy to the queue.")
    except ClientError as error:
        logger.exception("Couldn't add trust policy to the queue!")
        raise error


@staticmethod
def subscribe_queue_to_topic(topic, queue_arn):
    """
    Subscribe a queue to a topic.

    :param topic: The topic resource.
    :param queue_arn: The ARN of the queue.
    :return: The subscription resource.
    """
    try:
        subscription = topic.subscribe(
            Protocol="sqs",
            Endpoint=queue_arn,
        )
        logger.info("The queue is subscribed to the topic.")
        return subscription
    except ClientError as error:
        logger.exception("Couldn't subscribe queue to topic!")
        raise error


@staticmethod
def publish_price_update(topic, payload, group_id):
    """
    Compose and publish a message that updates the wholesale price.

    :param topic: The topic to publish to.
    :param payload: The message to publish.
    :param group_id: The group ID for the message.
    :return: The ID of the message.
    """
    try:
```

```
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- 有关API详细信息，请参阅 Python (Boto3) API 参考中的AWS SDK以下主题。
  - CreateTopic
  - Publish
  - 订阅

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建FIFO主题，在 Amazon SQS FIFO 队列中订阅该主题，然后向亚马逊SNS主题发布消息。

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                        "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
```

```
              iv_topicarn = lv_topic_arn
              iv_protocol = 'sqs'
              iv_endpoint = iv_queue_arn
          ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes
      ).
        ov_message_id = lo_result->get_messageid( ).                    "
 ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅中的以下主题AWS SDK以供SAPABAPAPI参考。
  - CreateTopic
  - Publish
  - 订阅

接收来自FIFO订阅的消息

现在，您可以在三个订阅的应用程序中接收价格更新。如中所示the section called “FIFO主题用例”，每个使用者应用程序的入口点都是 Amazon SQS 队列，其相应 AWS Lambda 函数可以自动轮询该队列。当 Amazon SQS 队列是 Lambda 函数的事件源时，Lambda 会根据需要扩展其轮询器队列以高效使用消息。

有关更多信息，请参阅《AWS Lambda 开发者指南》SQS中的AWS Lambda 与 Amazon 一起使用。有关编写自己的队列轮询器的信息，请参阅《亚马逊简单FIFO队列服务开发者指南》和《亚马逊简单队列服务API参考》ReceiveMessage中的 Amazon SQS 标准和队列建议。

使用 AWS CloudFormation

AWS CloudFormation 允许您使用模板文件将 AWS 资源集合一起创建和配置为一个单元。本部分提供的模板示例，用于创建以下内容：

- 发布价格SNSFIFO更新的 Amazon 主题
- 为批发和零售应用程序提供这些更新的Amazon SQS FIFO 队列
- 用于存储记录的分析应用程序的 Amazon SQS 标准队列，可以查询这些记录以获取商业智能 (BI)
- 将三个队列与主题关联的 Amazon SNS FIFO 订阅
- 指定订阅者应用程序的筛选策略只接收他们需要的价格更新

> **Note**
>
> 如果通过向主题发布消息来测试此代码示例，请确保您发布的是带 `business` 属性的消息。为属性值指定 `retail` 或 `wholesale`。否则，消息将被筛选掉，且不会传递到订阅的队列中。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
```

```
    "PriceUpdatesTopic": {
      "Type": "AWS::SNS::Topic",
      "Properties": {
        "TopicName": "PriceUpdatesTopic.fifo",
        "FifoTopic": true,
        "ContentBasedDeduplication": false,
        "ArchivePolicy": {
        "MessageRetentionPeriod": "30"
        }
      }
    },
    "WholesaleQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "WholesaleQueue.fifo",
        "FifoQueue": true,
        "ContentBasedDeduplication": false
      }
    },
    "RetailQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "RetailQueue.fifo",
        "FifoQueue": true,
        "ContentBasedDeduplication": false
      }
    },
    "AnalyticsQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "AnalyticsQueue"
      }
    },
    "WholesaleSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "WholesaleQueue",
            "Arn"
          ]
```

```
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false",
        "FilterPolicyScope": "MessageBody",
        "FilterPolicy": {
          "business": [
            "wholesale"
          ]
        }
      }
    },
    "RetailSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "RetailQueue",
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false",
        "FilterPolicyScope": "MessageBody",
        "FilterPolicy": {
          "business": [
            "retail"
          ]
        }
      }
    },
    "AnalyticsSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "AnalyticsQueue",
            "Arn"
          ]
```

```
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false"
      }
    },
    "SalesQueuesPolicy": {
      "Type": "AWS::SQS::QueuePolicy",
      "Properties": {
        "PolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "sns.amazonaws.com"
              },
              "Action": [
                "sqs:SendMessage"
              ],
              "Resource": "*",
              "Condition": {
                "ArnEquals": {
                  "aws:SourceArn": {
                    "Ref": "PriceUpdatesTopic"
                  }
                }
              }
            }
          ]
        },
        "Queues": [
          {
            "Ref": "WholesaleQueue"
          },
          {
            "Ref": "RetailQueue"
          },
          {
            "Ref": "AnalyticsQueue"
          }
        ]
      }
    }
  }
```

```
}
```

有关使用 AWS CloudFormation 模板部署 AWS 资源的更多信息，请参阅《AWS CloudFormation 用户指南》中的 “入门”。

Amazon SNS 邮件过滤

默认情况下，Amazon SNS 主题订阅者会收到发布到该主题的每条消息。要仅接收一部分消息，订阅者必须将筛选策略 分配给主题订阅。

过滤策略是一个包含属性的JSON对象，这些属性定义了订阅者将接收哪些消息。根据您为订阅设置的筛选策略范围，Amazon SNS 支持对消息属性或消息正文起作用的政策。消息正文的筛选策略假设消息负载是一个格式良好的JSON对象。

如果某个订阅没有筛选策略，则订阅者将接收发布到其主题的每条消息。当您向已设置筛选策略的主题发布消息时，Amazon 会将消息属性或消息正文与每个主题订阅的筛选策略中的属性进行SNS比较。如果任何消息属性或消息正文属性匹配，Amazon SNS 会将消息发送给订阅者。否则，Amazon SNS 不会向该订阅者发送消息。

有关更多信息，请参阅筛选发布到主题的消息。

主题

- Amazon SNS 订阅筛选政策范围
- 亚马逊SNS订阅筛选政策
- 在 Amazon 中应用订阅筛选政策 SNS
- 在 Amazon 中移除订阅筛选政策 SNS

Amazon SNS 订阅筛选政策范围

`FilterPolicyScope` 订阅属性允许您通过设置以下值之一来选择筛选范围：

- `MessageAttributes` – 筛选策略应用于消息属性。这是默认模式。
- `MessageBody` – 筛选策略应用于消息正文。

> **Note**
>
> 如果没有为现有筛选策略定义筛选策略范围，则范围默认为 `MessageAttributes`。

亚马逊SNS订阅筛选政策

订阅筛选策略允许您指定属性名称并向每个属性名称分配一个值列表。有关更多信息，请参阅 Amazon SNS 邮件过滤。

当 Amazon SNS 根据订阅筛选策略评估消息属性或消息正文属性时，它会忽略政策中未指定的属性或消息正文属性。

> **Important**
>
> AWS 诸如IAM和 Amazon 之类的服务SNS使用一种称为最终一致性的分布式计算模型。对订阅筛选器策略的添加或更改最多需要 15 分钟即可完全生效。

在以下条件下，一个订阅接受一条消息：

- 当筛选策略范围设置为 `MessageAttributes` 时，筛选策略中的每个属性名称都与消息属性名称相匹配。对于筛选策略中匹配的每个属性名称，至少有一个属性值与消息属性值相匹配。
- 当筛选策略范围设置为 `MessageBody` 时，筛选策略中的每个属性名称都与消息正文属性名称相匹配。对于筛选策略中匹配的每个属性名称，至少有一个属性值与消息正文属性值相匹配。

Amazon SNS 目前支持以下筛选运算符：

- AND亚马逊中的逻辑 SNS
- Amazon 中的 OR 逻辑 SNS
- Amazon 中的 OR 操作员 SNS
- Amazon 中的密钥匹配 SNS
- 数值精确匹配
- 数值 anything-but 匹配
- 数值范围匹配
- 字符串值精确匹配
- 字符串值 anything-but 匹配
- 使用前缀和 anything-but 运算符的字符串匹配
- 字符串值等于-忽略大小写
- 字符串值 IP 地址匹配

- 字符串值前缀匹配
- 字符串值后缀匹配

Amazon 筛选政策SNS示例

以下示例显示了由处理客户交易的 Amazon SNS 主题传送的消息负载。

第一个示例包括 MessageAttributes 字段，其中包含描述事务的属性：

- 客户的兴趣
- 存储名称
- 事件状态
- 购买价格 USD

由于此消息包含 MessageAttributes 字段，只要在订阅中将 FilterPolicyScope 设置为 MessageAttributes，任何设置了 FilterPolicy 的主题订阅都可以选择性地接受或拒绝消息。有关将属性应用于消息的信息，请参阅 Amazon SNS 消息属性。

```
{
   "Type": "Notification",
   "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
   "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
   "Message": "message-body-with-transaction-details",
   "Timestamp": "2019-11-03T23:28:01.631Z",
   "SignatureVersion": "4",
   "Signature": "signature",
   "UnsubscribeURL": "unsubscribe-url",
   "MessageAttributes": {
      "customer_interests": {
         "Type": "String.Array",
         "Value": "[\"soccer\", \"rugby\", \"hockey\"]"
      },
      "store": {
         "Type": "String",
         "Value":"example_corp"
      },
      "event": {
         "Type": "String",
         "Value": "order_placed"
      },
```

```
      "price_usd": {
         "Type": "Number",
         "Value": "210.75"
      }
   }
}
```

以下示例显示了 Message 字段中包含的相同属性，也称为消息有效负载或消息正文。只要在订阅中将 MessageBody 设置为 FilterPolicyScope，任何设置了 FilterPolicy 的主题订阅都可以选择性地接受或拒绝消息。

```
{
"Type": "Notification",
   "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
   "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
   "Message": "{
      \"customer_interests\": [\"soccer\", \"rugby\", \"hockey\"],
      \"store\": \"example_corp\",
      \"event\":\"order_placed\",
      \"price_usd\":210.75
   }",
   "Timestamp": "2019-11-03T23:28:01.631Z",
   "SignatureVersion": "4",
   "Signature": "signature",
   "UnsubscribeURL": "unsubscribe-url"
}
```

以下筛选策略基于消息的属性名称和值接受或拒绝消息。

接受示例消息的策略

以下订阅筛选策略中的属性与分配给示例消息的属性匹配。请注意，无论设置为 MessageAttributes 还是 MessageBody，相同的筛选策略都适用于 FilterPolicyScope。每个订阅者根据他们从主题收到的消息的构成来选择其筛选范围。

如果此策略中的任一属性与分配给该消息的属性不匹配，则此策略将拒绝该消息。

```
{
   "store": ["example_corp"],
   "event": [{"anything-but": "order_cancelled"}],
   "customer_interests": [
      "rugby",
```

```
      "football",
      "baseball"
   ],
   "price_usd": [{"numeric": [">=", 100]}]
}
```

拒绝示例消息的策略

以下订阅筛选策略在其属性与分配给示例消息的属性之间存在多个不匹配项。例如，由于消息属性中不存在 `encrypted` 属性名称，因此该策略属性会导致消息被拒绝，而不管分配给它的值如何。

如果存在任何不匹配项，则策略将拒绝消息。

```
{
   "store": ["example_corp"],
   "event": ["order_cancelled"],
   "encrypted": [false],
   "customer_interests": [
      "basketball",
      "baseball"
   ]
}
```

Amazon SNS 筛选政策限制

在为 Amazon SNS 订阅创建筛选策略时，了解政策中的密钥是如何计算的，这一点很重要。要记住的关键方面是：

1. 父密钥-父密钥是筛选策略中的顶级密钥。这些是您为其指定值或约束条件的密钥。
2. 属性名称-父密钥被视为筛选策略中的属性名称。您为这些键指定的值或约束条件将应用于消息负载中的相应属性。
3. 有效值-为父键指定的值必须是字符串、字符串数组或数字。如果该值是对象（例如，JSON对象），则在筛选策略中，它不会被视为有效密钥。

让我们考虑以下过滤策略示例：

```
{
  "state": ["SUCCESS"],
  "severity": [{ "exists": true }],
```

```
  "message": [{ "exists": true }],
  "finding": {
    "standard_control": [{ "exists": true }],
    "region": [{ "exists": true }],
    "account": [{ "exists": true }]
  }
}
```

在此示例中，以下密钥被计为筛选策略的一部分：

- state
- severity
- message
- standard_control
- region
- account

关键结果不计算在内，因为它包含一个JSON对象作为其值，而不是字符串、字符串数组或数字。

另一个示例是：

```
{
  "key_a": {
    "key_b": {
      "key_c": {
        "key_d": ["value_one", "value_two", "value_three", "value_four"]
      }
    },
    "key_e": {
      "key_f": ["value_one", "value_two", "value_three"]
    }
  }
}
```

在这种情况下，只有密钥key_d和key_f才算作筛选策略的一部分，因为它们分配的值可以是字符串或字符串数组。父键key_akey_b、和不计算key_c在内，因为它们包含嵌套JSON对象作为其值。

主题

- 常见策略限制

- 基于属性的筛选的策略限制
- 基于有效负载的筛选的策略限制

常见策略限制

- 字符串匹配-对于筛选策略中的字符串匹配，比较区分大小写。
- 数字匹配-对于数值匹配，该值的范围可以从 -10^9 到 10^9（-10 亿到 10 亿），小数点后有五位精度。
- 筛选策略复杂性-对于筛选策略的复杂性，值的总组合不得超过 150。要计算总组合，请将筛选策略中每个数组中的值数相乘。

考虑以下策略示例：

```
{
   "key_a": ["value_one", "value_two", "value_three"],
   "key_b": ["value_one"],
   "key_c": ["value_one", "value_two"]
}
```

在本政策中：

- 第一个数组有 3 个值
- 第二个数组有 1 个值
- 第三个数组有 2 个值

总组合数的计算方法如下所示：

- 3 x 1 x 2 = 6

筛选策略语法

筛选策略JSON中可以包含以下内容：

- 用引号引起来的字符串
- 数字
- 不带引号的关键字 `true`、`false` 和 `null`

使用 Amazon 时 SNSAPI，您必须将JSON筛选策略中的作为有效的 UTF -8 字符串传递。

筛选策略限制

- 筛选策略的最大大小为 256 KB。
- 默认情况下，每个主题最多可以有 200 个筛选策略，每个 AWS 账户最多可以有 10,000 个筛选策略。
- 此政策限制不会阻止使用创建 Amazon SQS 队列订阅SubscribeAPI。但是，当您在SubscribeAPI呼叫（或呼SetSubscriptionAttributesAPI叫）中附加过滤策略时，它将失败。
- 要增加此限额，您可以使用 AWS 服务限额。

基于属性的筛选的策略限制

- 基于属性的筛选是默认选项。`FilterPolicyScope` 在订阅中设置为 `MessageAttributes`。
- Amazon SNS 不接受基于属性的筛选的嵌套筛选策略。
- Amazon 仅SNS将策略属性与具有以下数据类型的消息属性进行比较：
  - `String`
  - `String.Array`

    > **Important**
    >
    > 不建议在数组中传递对象，因为基于属性的筛选不支持嵌套，可能会产生意想不到的结果。对嵌套策略使用基于有效载荷的筛选。

  - `Number`
- Amazon SNS 会忽略`Binary`数据类型的消息属性。
- 一个筛选策略最多可具有 5 个属性名称。

基于有效负载的筛选的策略限制

Amazon SNS 接受基于有效负载的筛选的嵌套筛选策略。要计算筛选策略中值的总组合，请将每个嵌套数组中的值数相乘。

考虑以下策略示例：

```
{
"key_a": {
    "key_b": {
        "key_c": ["value_one", "value_two", "value_three", "value_four"]
        }
    },
"key_d": {
    "key_e": ["value_one", "value_two", "value_three"]
    }
}
```

在本政策中：

- 第一个数组在三级嵌套键中有四个值。
- 第二个在两级嵌套键中包含三个值。

总组合数的计算方法如下所示：

- 4 x 3 x 3 x 2 = 72

政策限制

一个筛选策略最多可以有五个父密钥（顶级密钥）。对于嵌套策略，只有父密钥才计入五个密钥的限制。

数值范围

对于筛选策略中的数字匹配，该值的范围可以从 -10^9 到 10^9（-10 亿到 10 亿），小数点后精度为五位数。

切换到基于有效负载的筛选

要从基于属性（默认）的筛选切换到基于有效负载的筛选，您必须在订阅中将 `FilterPolicyScope` 设置为 `MessageBody`。

AND亚马逊中的 /OR 逻辑 SNS

您可以使用包含 AND /OR 逻辑的操作来匹配消息属性或消息正文属性。

主题

- AND亚马逊中的逻辑 SNS

- Amazon 中的 OR 逻辑 SNS
- Amazon 中的 OR 操作员 SNS

AND亚马逊中的逻辑 SNS

您可以使用多个属性名称来应用AND逻辑。

考虑以下策略：

```
{
  "customer_interests": ["rugby"],
  "price_usd": [{"numeric": [">", 100]}]
}
```

它匹配任何 customer_interests 值设置为 rugby 且 price_usd 值设置为大于 100 的数字的消息属性或消息正文属性。

> **Note**
>
> 您不能将AND逻辑应用于同一属性的值。

Amazon 中的 OR 逻辑 SNS

通过将多个值分配给一个属性名称来应用 OR 逻辑。

考虑以下策略：

```
{
   "customer_interests": ["rugby", "football", "baseball"]
}
```

它匹配任何 customer_interests 值设置为 rugby、football、或 baseball 的消息属性或消息正文属性。

Amazon 中的 OR 操作员 SNS

您可以使用 "$or" 运算符明确定义筛选策略，以表达策略中多个属性之间的 OR 关系。

SNS只有当政策满足以下所有条件时，Amazon 才会识别该"$or"关系。当其中任一条件未满足时，"$or" 会被视为常规属性名称，与策略中的任何其他字符串相同。

- 规则中有一个 "$or" 字段属性，后跟一个数组，例如，“$or” : []。
- "$or" 数组中至少有 2 个对象："$or": [{}, {}]。
- "$or" 数组中的所有对象都没有属于保留关键字的字段名。

否则，"$or" 将被视为普通属性名称，与策略中的其他字符串相同。

由于数字和前缀是保留关键字，因此以下策略不会被解析为 OR 关系。

```
{
   "$or": [ {"numeric" : 123}, {"prefix": "abc"} ]
}
```

OR 运算符示例

标准 OR：

```
{
  "source": [ "aws.cloudwatch" ],
  "$or": [
    { "metricName": [ "CPUUtilization" ] },
    { "namespace": [ "AWS/EC2" ] }
  ]
}
```

此策略的筛选逻辑是：

```
"source" && ("metricName" || "namespace")
```

它匹配以下任一组消息属性：

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
"metricName": {"Type": "String", "Value": "CPUUtilization"}
```

或者

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
"namespace": {"Type": "String", "Value": "AWS/EC2"}
```

它还匹配以下任一消息正文：

```
{
    "source": "aws.cloudwatch",
    "metricName": "CPUUtilization"
}
```

或者

```
{
    "source": "aws.cloudwatch",
    "namespace": "AWS/EC2"
}
```

包括 OR 关系在内的策略限制

考虑以下策略：

```
{
    "source": [ "aws.cloudwatch" ],
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      {
        "metricType": [ "MetricType" ] ,
        "$or" : [
          { "metricId": [ 1234, 4321 ] },
          { "spaceId": [ 1000, 2000, 3000 ] }
        ]
      }
    ]
  }
```

该策略的逻辑也可以简化为：

```
("source" AND "metricName")
OR
("source" AND "metricType" AND "metricId")
OR
("source" AND "metricType" AND "spaceId")
```

具有 OR 关系的策略的复杂度计算可以简化为每个 OR 语句的组合复杂度之和。

总组合数的计算方法如下所示：

```
(source * metricName) + (source * metricType * metricId) + (source * metricType *
 spaceId)
= (1 * 2) + (1 * 1 * 2) + (1 * 1 * 3)
= 7
```

source 有一个值，metricName 有两个值，metricType 有一个值，metricId 有两个值，spaceId 有三个值。

考虑以下嵌套筛选策略：

```
{
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      { "namespace": [ "AWS/EC2", "AWS/ES" ] }
    ],
    "detail" : {
      "scope" : [ "Service" ],
      "$or": [
        { "source": [ "aws.cloudwatch" ] },
        { "type": [ "CloudWatch Alarm State Change"] }
      ]
    }
  }
```

该策略的逻辑可以简化为：

```
("metricName" AND ("detail"."scope" AND "detail"."source")
OR
("metricName" AND ("detail"."scope" AND "detail"."type")
OR
("namespace" AND ("detail"."scope" AND "detail"."source")
OR
("namespace" AND ("detail"."scope" AND "detail"."type")
```

组合总数的计算方法与非嵌套策略相同，只是我们需要考虑键的嵌套级别。

总组合数的计算方法如下所示：

```
(2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) = 32
```

metricName 有两个值，namespace 有两个值，scope 是具有一个值的两级嵌套键，source 是具有一个值的两级嵌套键，type 是具有一个值的两级嵌套键。

Amazon 中的密钥匹配 SNS

您可以使用 exists 运算符来匹配在筛选策略中具有或不具有指定属性的传入消息。exists 匹配只对叶节点有效。它对于中间节点不起作用。

- 使用 "exists": true 匹配包含指定属性的传入消息。键值必须为非空值。

  例如，以下策略属性使用值为 true 的 exists 运算符：

  ```
  "store": [{"exists": true}]
  ```

  它匹配包含 store 属性键的任何消息属性列表，如下所示：

  ```
  "store": {"Type": "String", "Value": "fans"}
  "customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
  \"]"}
  ```

  它还匹配以下任一消息正文：

  ```
  {
      "store": "fans"
      "customer_interests": ["baseball", "basketball"]
  }
  ```

  但是，它不匹配不含 store 属性键的任何消息属性列表，如下所示：

  ```
  "customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
  \"]"}
  ```

  它与以下消息正文也不匹配：

  ```
  {
      "customer_interests": ["baseball", "basketball"]
  }
  ```

- 使用 "exists": false 匹配不 包含指定属性的传入消息。

> **Note**
>
> `"exists": false` 仅在存在至少一个属性时才匹配。一组空的属性会导致筛选条件不匹配。

例如，以下策略属性使用值为 `false` 的 `exists` 运算符：

```
"store": [{"exists": false}]
```

它不 匹配包含 `store` 属性键的任何消息属性列表，如下所示：

```
"store": {"Type": "String", "Value": "fans"}
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

它与以下消息正文也不匹配：

```
{
    "store": "fans"
    "customer_interests": ["baseball", "basketball"]
}
```

但是，它匹配不含 `store` 属性键的任何消息属性列表，如下所示：

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

它还匹配以下消息正文：

```
{
    "customer_interests": ["baseball", "basketball"]
}
```

在 Amazon 中匹配数值 SNS

您可以通过将数值与消息属性值或消息正文属性值进行匹配来筛选消息。在JSON策略中，数值不用双引号括起来。您可以使用以下数值运算进行筛选。

> **Note**
>
> 前缀仅支持字符串 匹配。

主题

在 Amazon 中精确匹配 SNS

当策略属性值包含 `numeric` 关键字和 `=` 运算符时，它将匹配具有相同名称和相同数值的任何消息属性值或消息正文属性值。

请考虑以下策略属性：

```
"price_usd": [{"numeric": ["=",301.5]}]
```

它匹配以下任一消息属性：

```
"price_usd": {"Type": "Number", "Value": 301.5}
```

```
"price_usd": {"Type": "Number", "Value": 3.015e2}
```

它还匹配以下任一消息正文：

```
{
   "price_usd": 301.5
}
```

```
{
```

```
   "price_usd": 3.015e2
}
```

除了在 Amazon 中匹配之外什么都没有 SNS

当策略属性值包含关键字 anything-but 时，它会匹配不 包含任何策略属性值的任何消息属性值或消息正文属性值。

请考虑以下策略属性：

```
"price": [{"anything-but": [100, 500]}]
```

它匹配以下任一消息属性：

```
"price": {"Type": "Number", "Value": 101}
```

```
"price": {"Type": "Number", "Value": 100.1}
```

它还匹配以下任一消息正文：

```
{
   "price": 101
}
```

```
{
   "price": 100.1
}
```

此外，它还匹配以下消息属性（因为它包含的值不是 100 或 500）：

```
"price": {"Type": "Number.Array", "Value": "[100, 50]"}
```

它还匹配以下消息正文（因为它包含的值不是 100 或 500）：

```
{
   "price": [100, 50]
}
```

但是，它不匹配以下消息属性：

```
"price": {"Type": "Number", "Value": 100}
```

它与以下消息正文也不匹配：

```
{
   "price": 100
}
```

值范围匹配

除了 = 运算符之外，数值策略属性还可以包含以下运算符：<、<=、> 和 >=。

请考虑以下策略属性：

```
"price_usd": [{"numeric": ["<", 0]}]
```

它匹配任何具有负数值的消息属性或消息正文属性。

考虑另一个消息属性：

```
"price_usd": [{"numeric": [">", 0, "<=", 150]}]
```

它匹配任何具有正数（最大为 150）的消息属性或消息正文属性。

在 Amazon 中匹配字符串值 SNS

您可以通过将字符串值与消息属性值或消息正文属性值进行匹配来筛选消息。在JSON策略中，字符串值用双引号括起来。您可以使用以下字符串操作来匹配消息属性或消息正文。

主题

- 精确匹配
- Anything-but 匹配
- 将前缀与 anything-but 运算符结合使用
- E quals-ignore-case 匹配
- IP 地址匹配
- 前缀匹配
- 后缀匹配

精确匹配

在策略属性值与一个或多个消息属性值匹配时，会进行精确匹配。

请考虑以下策略属性：

```
"customer_interests": ["rugby", "tennis"]
```

它匹配以下消息属性：

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

```
"customer_interests": {"Type": "String", "Value": "tennis"}
```

它还匹配以下消息正文：

```
{
   "customer_interests": "rugby"
}
```

```
{
   "customer_interests": "tennis"
}
```

但是，它不匹配以下消息属性：

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

它与以下消息正文也不匹配：

```
{
   "customer_interests": "baseball"
}
```

Anything-but 匹配

当策略属性值包含关键字 anything-but 时，它会匹配不 包含任何策略属性值的任何消息属性值或消息正文值。anything-but 可以与 "exists": false 组合。

请考虑以下策略属性：

```
"customer_interests": [{"anything-but": ["rugby", "tennis"]}]
```

它匹配以下任一消息属性：

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "football"}
```

它还匹配以下任一消息正文：

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "football"
}
```

此外，它还匹配以下消息属性（因为它包含的值不是 rugby 或 tennis）：

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"baseball\"]"}
```

它还匹配以下消息正文（因为它包含的值不是 rugby 或 tennis）：

```
{
   "customer_interests": ["rugby", "baseball"]
}
```

但是，它不匹配以下消息属性：

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

它与以下消息正文也不匹配：

```
{
   "customer_interests": ["rugby"]
```

```
}
```

将前缀与 anything-but 运算符结合使用

要进行字符串匹配，您也可以将前缀与 anything-but 运算符结合使用。例如，以下策略属性拒绝 order- 前缀：

```
"event":[{"anything-but": {"prefix": "order-"}}]
```

它匹配以下任一属性：

```
"event": {"Type": "String", "Value": "data-entry"}
```

```
"event": {"Type": "String", "Value": "order_number"}
```

它还匹配以下任一消息正文：

```
{
   "event": "data-entry"
}
```

```
{
   "event": "order_number"
}
```

但是，它不匹配以下消息属性：

```
"event": {"Type": "String", "Value": "order-cancelled"}
```

它与以下消息正文也不匹配：

```
{
   "event": "order-cancelled"
}
```

E quals-ignore-case 匹配

当策略属性包含关键字 equals-ignore-case 时，它将与任何信息属性或正文属性值进行不区分大小写的匹配。

请考虑以下策略属性：

```
"customer_interests": [{"equals-ignore-case": "tennis"}]
```

它匹配以下任一消息属性：

```
"customer_interests": {"Type": "String", "Value": "TENNIS"}
```

```
"customer_interests": {"Type": "String", "Value": "Tennis"}
```

它还匹配以下任一消息正文：

```
{
    "customer_interests": "TENNIS"
}
```

```
{
    "customer_interests": "teNnis"
{
```

IP 地址匹配

您可以使用 cidr 运算符来检查传入消息是否源自特定 IP 地址或子网。

请考虑以下策略属性：

```
"source_ip":[{"cidr": "10.0.0.0/24"}]
```

它匹配以下任一消息属性：

```
"source_ip": {"Type": "String", "Value": "10.0.0.0"}
```

```
"source_ip": {"Type": "String", "Value": "10.0.0.255"}
```

它还匹配以下任一消息正文：

```
{
   "source_ip": "10.0.0.0"
```

```
}
```

```
{
   "source_ip": "10.0.0.255"
}
```

但是，它不匹配以下消息属性：

```
"source_ip": {"Type": "String", "Value": "10.1.1.0"}
```

它与以下消息正文也不匹配：

```
{
   "source_ip": "10.1.1.0"
}
```

前缀匹配

当策略属性值包含关键字 `prefix` 时，它匹配以指定字符开头的任何消息属性值或正文属性值。

请考虑以下策略属性：

```
"customer_interests": [{"prefix": "bas"}]
```

它匹配以下任一消息属性：

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

它还匹配以下任一消息正文：

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "basketball"
}
```

但是，它不匹配以下消息属性：

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

它与以下消息正文也不匹配：

```
{
   "customer_interests": "rugby"
}
```

后缀匹配

当策略属性值包含关键字 `suffix` 时，它匹配以指定字符结尾的任何消息属性值或正文属性值。

请考虑以下策略属性：

```
"customer_interests": [{"suffix": "ball"}]
```

它匹配以下任一消息属性：

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

它还匹配以下任一消息正文：

```
{
    "customer_interests": "baseball"
}
```

```
{
    "customer_interests": "basketball"
}
```

但是，它不匹配以下消息属性：

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

它与以下消息正文也不匹配：

```
{
    "customer_interests": "rugby"
}
```

在 Amazon 中应用订阅筛选政策 SNS

Amazon 中的邮件筛选SNS允许您根据筛选策略有选择地向订阅者传送消息。这些策略定义了发送到订阅的消息必须满足的条件。虽然原始消息传送是一个可能影响消息处理的选项，但订阅过滤器并不需要它才能起作用。

您可以使用亚马逊SNS控制台将筛选策略应用于亚马逊SNS订阅。或者，要以编程方式应用政策，您可以使用 Amazon SNS API、 AWS Command Line Interface (AWS CLI) 或任何支持 Amazon AWS SDK SNS 的策略。你也可以使用 AWS CloudFormation。

启用原始消息传送

原始消息交付可确保消息负载按原样交付给订阅者，而无需进行任何额外的编码或转换。当订阅者需要使用原始消息格式进行处理时，这可能很有用。但是，原始消息传递与订阅过滤器的功能没有直接关系。

应用订阅筛选器

要将邮件过滤器应用于订阅，请使用JSON语法定义过滤器策略。该政策规定了消息必须满足的条件才能发送到订阅中。过滤器可以基于消息属性，例如消息属性、消息结构甚至消息内容。

原始消息传送和订阅过滤器之间的关系

虽然启用原始消息传送会影响订阅者传递和处理消息的方式，但这并不是使用订阅过滤器的先决条件。但是，在订阅者需要不做任何修改的原始消息格式的情况下，启用原始消息传送可能与订阅过滤器一起启用。

有效筛选的注意事项

在实现消息过滤时，请考虑您的应用程序和订阅者的特定要求。定义与邮件传送标准精确匹配的过滤策略，以确保高效、有针对性的消息分发。

> **Important**
>
> AWS 诸如IAM和 Amazon 之类的服务SNS使用一种称为最终一致性的分布式计算模型。对订阅筛选器策略的添加或更改最多需要 15 分钟即可完全生效。

AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择 Subscriptions（订阅）。
3. 选择订阅，然后选择编辑。
4. 在 Edit（编辑）页面上，展开 Subscription filter policy（订阅筛选策略）部分。
5. 在 attribute-based filtering（基于属性的筛选）或 payload-based filtering（基于有效负载的筛选）之间进行选择。
6. 在JSON编辑器字段中，提供您的筛选策略的JSON正文。
7. 选择保存更改。

   Amazon 将您的筛选政策SNS应用于订阅。

AWS CLI

要使用 AWS Command Line Interface (AWS CLI) 应用筛选策略，请使用set-subscription-attributes命令，如以下示例所示。对于 `--attribute-name` 选项，请指定 `FilterPolicy`。对于`--attribute-value`，请指定您的JSON策略。

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --
attribute-name FilterPolicy --attribute-value '{"store":["example_corp"],"event":
["order_placed"]}'
```

要使您的策略有效，请JSON用双引号将属性名称和值括起来。此外，您必须用引号将整个策略参数括起来。为避免转义引号，您可以使用单引号将策略括起来，使用双引号将JSON名称和值括起来，如上面的示例所示。

如果要从基于属性（默认）的邮件筛选切换到基于负载的邮件过滤，也可以使用该set-subscription-attributes命令。对于 `--attribute-name` 选项，请指定 `FilterPolicyScope`。对于 `--attribute-value`，请指定 `MessageBody`。

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-
name FilterPolicyScope --attribute-value MessageBody
```

要验证是否已应用您的筛选策略，请使用 `get-subscription-attributes` 命令。终端输出中的属性应显示 `FilterPolicy` 键的筛选策略，如以下示例中所示：

```
$ aws sns get-subscription-attributes --subscription-arn arn:aws:sns: ...
{
    "Attributes": {
        "Endpoint": "endpoint . . .",
        "Protocol": "https",
        "RawMessageDelivery": "false",
        "EffectiveDeliveryPolicy": "delivery policy . . .",
        "ConfirmationWasAuthenticated": "true",
        "FilterPolicy": "{\"store\": [\"example_corp\"], \"event\": [\"order_placed
\"]}",
        "FilterPolicyScope": "MessageAttributes",
        "Owner": "111122223333",
        "SubscriptionArn": "arn:aws:sns: . . .",
        "TopicArn": "arn:aws:sns: . . ."
    }
}
```

AWS SDKs

以下代码示例演示如何使用 SetSubscriptionAttributes。

> **Important**
>
> 如果您使用的是 For SDK Java 2.x 示例，则该类SNSMessageFilterPolicy不是开箱即用的。有关如何安装该类的说明，请参阅 GitHub 网站上的示例。

CLI

AWS CLI

设置订阅属性

以下set-subscription-attributes示例将该RawMessageDelivery属性设置为订SQS阅。

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

此命令不生成任何输出。

以下set-subscription-attributes示例为SQS订阅设置一个FilterPolicy属性。

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

此命令不生成任何输出。

以下set-subscription-attributes示例从SQS订阅中删除了该FilterPolicy属性。

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

此命令不生成任何输出。

- 有关API详细信息，请参阅AWS CLI 命令参考SetSubscriptionAttributes中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");
```

```
            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 SetSubscriptionAttributes” 中的。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
```

```
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- 有关API详细信息，请参阅SetSubscriptionAttributes中的 AWS SDKPython (Boto3) API 参考。

亚马逊 SNS API

要向 Amazon 应用筛选政策 SNSAPI，请向该SetSubscriptionAttributes操作提出请求。将`AttributeName`参数设置为`FilterPolicy`，然后将`AttributeValue`参数设置为筛选策略 JSON。

如果要从基于属性（默认）的消息筛选切换到基于有效负载的消息筛选，您也可以使用 SetSubscriptionAttributes 操作。将 AttributeName 参数设置为 FilterPolicyScope，并将 AttributeValue 参数设置为 MessageBody。

AWS CloudFormation

要使用应用筛选策略 AWS CloudFormation，请使用JSON或YAML模板创建 AWS CloudFormation 堆栈。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的AWS::SNS::Subscription资源FilterPolicy属性和示例 AWS CloudFormation 模板。

1. 登录 AWS CloudFormation 控制台。
2. 选择创建堆栈。
3. 在 Select Template (选择模板) 页面上，依次选择 Upload a template to Amazon S3 (将模板上传到 Amazon S3)、您的文件和下一步。
4. 在指定详细信息页面中，执行以下操作：
   a. 对于堆栈名称，键入 MyFilterPolicyStack。
   b. 对于 myHttpEndpoint，键入要订阅您的主题的HTTP终端节点。

   > **Tip**
   > 如果您没有HTTP终端节点，请创建一个。

5. 在选项页面上，选择下一步。
6. 在 Review 页面上，选择 Create 。

在 Amazon 中移除订阅筛选政策 SNS

要停止筛选发送到订阅的消息，请使用空JSON正文覆盖该订阅的过滤策略，从而将其删除。在删除该策略后，订阅会接受发布到它的每条消息。

AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择 Subscriptions（订阅）。
3. 选择订阅，然后选择编辑。

4. 在 E dit 中 ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***页面上，展开 “订阅筛选策略” 部分。
5. 在JSON编辑器字段中，为您的过滤器策略提供一个空白的JSON正文：{}。
6. 选择 Save changes（保存更改）。

   Amazon 将您的筛选政策SNS应用于订阅。

AWS CLI

要使用删除过滤器策略 AWS CLI，请使用set-subscription-attributes命令并为--attribute-value参数提供一个空的JSON正文：

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-name FilterPolicy --attribute-value "{}"
```

Amazon SNS API

要移除 Amazon 的筛选政策 SNSAPI，请提出SetSubscriptionAttributes操作请求。将AttributeName参数设置为FilterPolicy，并为该AttributeValue参数提供一个空的JSON正文。

Amazon 中的消息数据保护 SNS

主题

什么是消息数据保护？

消息数据保护使用数据保护策略来审计、屏蔽、编辑或屏蔽在应用程序或 AWS 服务之间移动的敏感信息，从而保护发布到您的 Amazon SNS 主题的数据。

消息数据保护使用数据标识符扫描动态数据中的个人身份信息 (PII) 和受保护的健康信息 (PHI)。您可以选择使用预定义的（或亚马逊SNS托管的）数据标识符（例如姓名、地址、信用卡号和处方药代码），也可以根据您的业务用例创建自己的自定义数据标识符。使用扫描的信息，消息数据保护功能提供详细的审计日志，并允许您采取措施来保护该数据。

消息数据保护功能支持以下操作，以帮助保护敏感的客户信息：

- 审计 — 对发布到亚马逊SNS主题的数据中，最多可审计 99% 的数据。然后，您可以选择将调查结果发送到亚马逊 CloudWatch、亚马逊 S3 或亚马逊 Data Firehose。
- 去身份识别 – 遮蔽或去除敏感数据而不中断消息发布或传输。
- 拒绝-如果有效负载中存在敏感数据，则阻止应用程序和 AWS 资源之间的数据传输。

> **Note**
>
> 亚马逊仅SNS支持亚马逊SNS标准主题的消息数据保护。

为什么应该使用消息数据保护？

通过在监管、风险管理和合规计划中引入消息数据保护功能，您可以实施数据保护策略来帮助识别和防止数据泄露。这为您的团队提供了工具，可通过遵守隐私法规（例如、、和美联储）来帮助降低财务

HIPAAGDPRPCI、法律和监管风险RAMP。它还可以让您的开发人员摆脱为了保护敏感数据而构建和管理自己的工具的相关运营开销。

例如，您可以使用消息数据保护功能创建审计策略，确定是否有任何系统无意中发送或接收了敏感数据。如果您的审计结果表明系统将信用卡信息发送到了无需这些信息的系统，您可以使用阻止策略 来阻止传输这些数据。

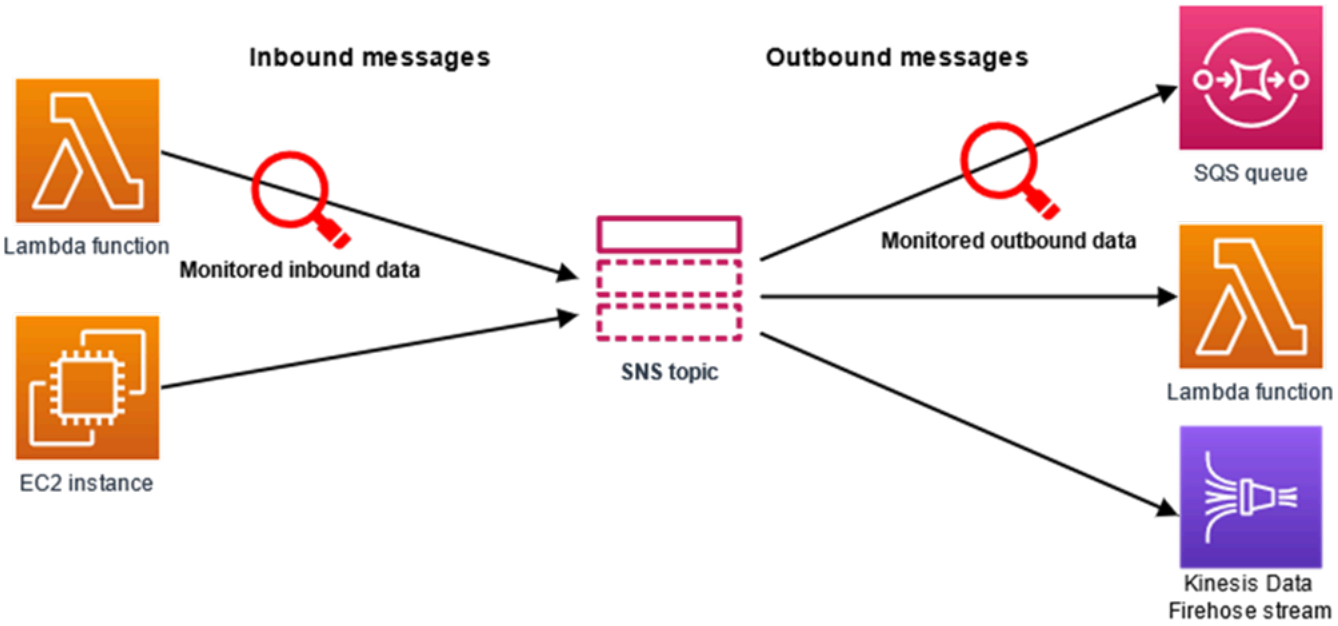
> **Note**
>
> 亚马逊仅SNS支持亚马逊SNS标准主题的消息数据保护。

了解亚马逊的数据保护政策 SNS

主题

什么是数据保护策略？

Amazon SNS 使用数据保护策略来选择您要扫描的敏感数据，以及您要采取哪些措施来保护这些数据不被您的亚马逊SNS主题交换。要选择感兴趣的敏感数据，您可以使用数据标识符。然后，Amazon SNS 消息数据保护使用机器学习和模式匹配来检测敏感数据。要对找到的数据标识符采取操作，您可以定义审计、去身份识别或拒绝操作。利用这些操作，您可以记录已找到（或未找到）的敏感数据，遮蔽或者去除敏感数据，或者拒绝消息传送。

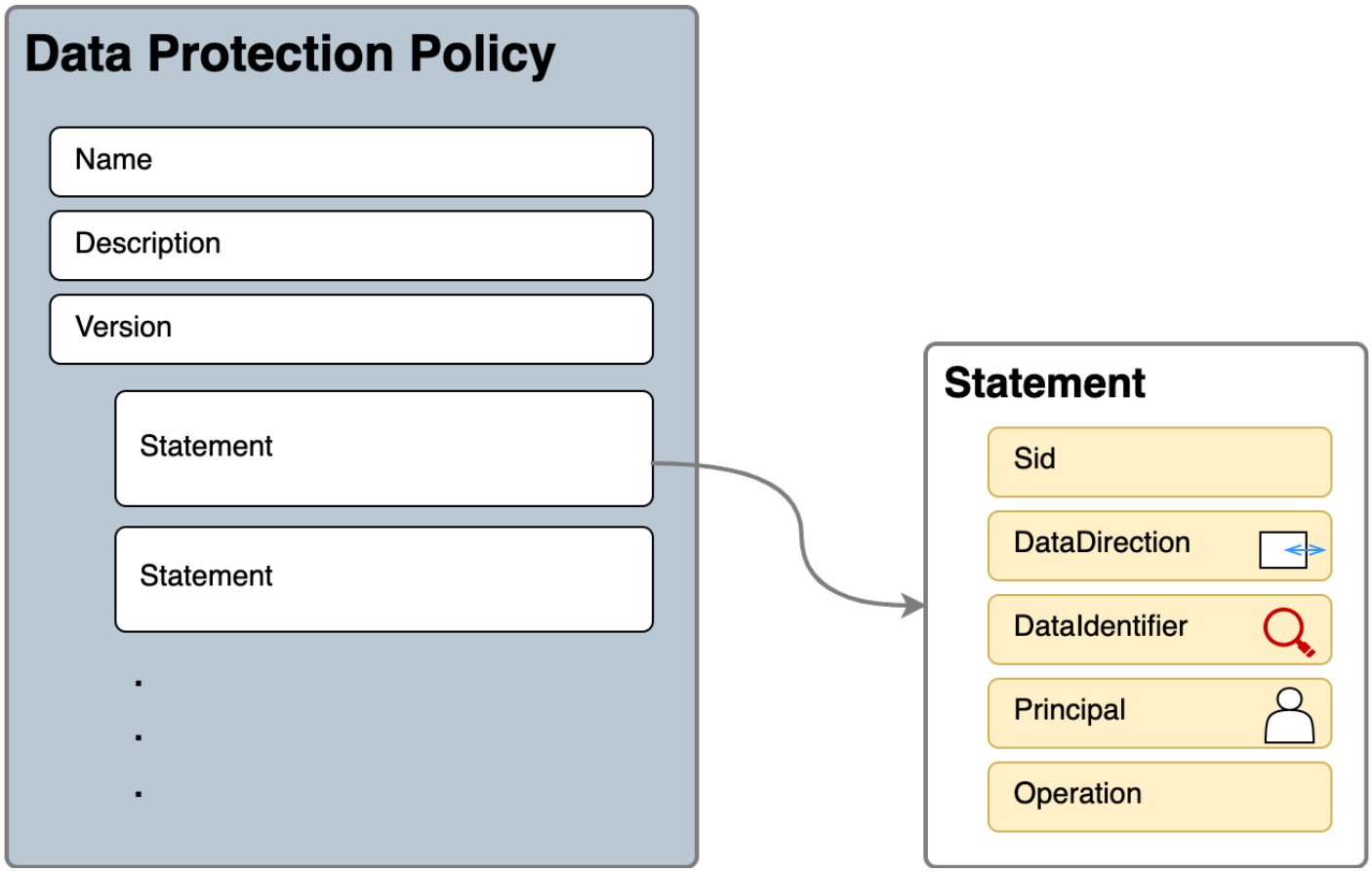


数据保护策略采用什么结构？

如下图所示，数据保护策略文档包含以下元素：

- 文档顶部的可选策略范围信息
- 一个或多个单独语句

每个语句都包含有关单个权限的信息。



每个 Amazon SNS 主题只能定义一项数据保护政策。数据保护策略可以有一个或多个拒绝或去身份识别语句，但只能有一个审计语句。

JSON数据保护策略的属性

数据保护策略需要以下基本策略信息用于识别：

- Name（名称）– 策略名称。
- Description（描述）(可选)– 策略描述。
- Version（版本）– 策略语言版本。当前版本为 2021-06-01。
- Statement（语句）– 指定数据保护策略操作的语句列表。

```
{
  "Name": "basicPII-protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
  "Statement": [
        ...
```

```
  ]
}
```

JSON策略声明的属性

策略语句设置数据保护操作的检测上下文。

- Sid（可选）– 语句标识符。
- DataDirection— 与 Amazon SNS 主题相关的入库（用于发布API请求）或出库（用于通知传送）。
- DataIdentifier— Amazon SNS 主题应扫描的敏感数据。例如，姓名、地址或电话号码。
- 校长 — IAM 针对该主题发布的委IAM托人，或订阅该主题的委托人。
- 操作 — 后续操作，即 “审计”、“取消识别”（屏蔽或隐藏）或 “拒绝”（阻止），Amazon SNS 主题在发现敏感数据后就会执行这些操作。

```
{
    "Sid": "basicPII-inbound-protection",
    "DataDirection": "Inbound",
    "Principal": ["*"],
    "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Name",
        "arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US"
    ],
    "Operation": {
        ...
    }
}
```

JSON策略声明操作的属性

策略语句设置以下数据保护操作之一。

- Audit（审计）– 发布指标并发现结果日志，而不中断消息发布或传输。
- De-identify（去身份识别）– 遮蔽或去除敏感数据而不中断消息发布。
- 拒绝 — 阻止 Amazon SNS 发布请求或消息传送失败。

如何确定我的数据保护政策的IAM委托人？

消息数据保护使用两个与 Amazon SNS 交互的IAM委托人。

1. 发布API委托人（入站）-调用 Amazon 的经过身份验证的IAM委托人SNS`Publish`API。
2. 订阅委托人（出站）-订阅创建`Subscribe`API期间调用的经过身份验证的IAM委托人。

`SubscriptionPrincipal`是可公开获得的 Amazon SNS 订阅资产，可从中检索`GetSubscriptionAttributes`API。

```
{
  "Attributes": {
    "SubscriptionPrincipal": "arn:aws:iam::123456789012:user/NoNameAccess",
    "Owner": "123412341234",
    "RawMessageDelivery": "true",
    "TopicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
    "Endpoint": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
    "Protocol": "sqs",
    "PendingConfirmation": "false",
    "ConfirmationWasAuthenticated": "true",
    "SubscriptionArn": "arn:aws:sns:us-east-1:123412341234:PII-data-
topic:5d8634ef-67ef-49eb-a824-4042b28d6f55"
  }
}
```

亚马逊的数据保护政策运作 SNS

以下数据保护策略示例可用于审计和拒绝敏感数据。有关包含示例应用程序的完整教程，请参阅 Amazon 消息数据保护简介SNS博客文章。

主题

- 审计操作
- 去身份识别操作
- 拒绝操作

审计操作

审计操作对主题入站消息进行采样，并记录 AWS 目标中发现的敏感数据。采样率可以是 0-99 之间的整数。此操作需要以下类型的日志记录目标之一：

1. FindingsDestination— Amazon SNS 主题在有效负载中发现敏感数据时的日志记录目标。
2. NoFindingsDestination— Amazon SNS 主题在有效负载中找不到敏感数据时的日志目标。

您可以在以下每种日志目标类型 AWS 服务 中使用以下内容：

- Amaz CloudWatch on Logs（可选）— LogGroup 必须位于主题区域中，并且名称必须以 /aws/vendedlogs/ 开头。
- Amazon Data Firehose（可选）— DeliveryStream 必须位于主题区域中，并且必须将 Di rec PUT t 作为传送流的来源。有关更多详情，请参阅 Amazon Data Firehose 开发者指南中的来源、目标和名称。
- Amazon S3（可选）– Amazon S3 存储桶名称。要@@ 使用启用了KMS加密功能的 Amazon S3 存储桶，SSE则需要执行额外操作。

```
{
  "Operation": {
    "Audit": {
      "SampleRate": "99",
      "FindingsDestination": {
            "CloudWatchLogs": {
                "LogGroup": "/aws/vendedlogs/log-group-name"
            },
            "Firehose": {
                "DeliveryStream": "delivery-stream-name"
            },
            "S3": {
                "Bucket": "bucket-name"
            }
      },
      "NoFindingsDestination": {
            "CloudWatchLogs": {
                "LogGroup": "/aws/vendedlogs/log-group-name"
            },
            "Firehose": {
                "DeliveryStream": "delivery-stream-name"
            },
            "S3": {
                "Bucket": "bucket-name"
            }
      }
    }
  }
}
```

指定日志目标时所需的权限

在数据保护策略中指定日志记录目标时，必须将以下权限添加到调用 Amazon 的IAM委托IAM人的身份策略中 SNS `PutDataProtectionPolicy`API，或者`CreateTopic`API使用`--data-protection-policy`参数。

审计目标	IAM许可
默认	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy 要@@ 使用启用了KMS加密功能的 Amazon S3 存储桶，SSE则需要执行额外操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs:DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutResourcePolicy",
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:region:account-id:SampleLogGroupName:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "firehose:TagDeliveryStream"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutBucketPolicy",
        "s3:GetBucketPolicy"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ]
    }
  ]
}
```

与 SSE-一起使用的必需密钥策略 KMS

如果您使用 Amazon S3 存储桶作为日志目标，则可以通过使用 Amazon S3 托管密钥启用服务器端加密 (SSE-S3) 或使用 (-) 启用服务器端加密来保护存储桶中的数据。 AWS KMS keys SSE KMS有关详情，请参阅《Amazon S3 用户指南》中的使用服务器端加密保护数据。

如果选择 SSE-S3，则无需进行其他配置。Amazon S3 处理加密密钥。

如果您选择 SSE-KMS，则必须使用客户管理的密钥。您必须更新客户托管密钥的密钥策略，以便日志传输账户可以写入 S3 存储桶。有关与 SSE-一起使用的所需密钥策略的更多信息KMS，请参阅《亚马逊 CloudWatch 日志用户指南》中的 Amazon S3 存储桶服务器端加密。

审计目标日志示例

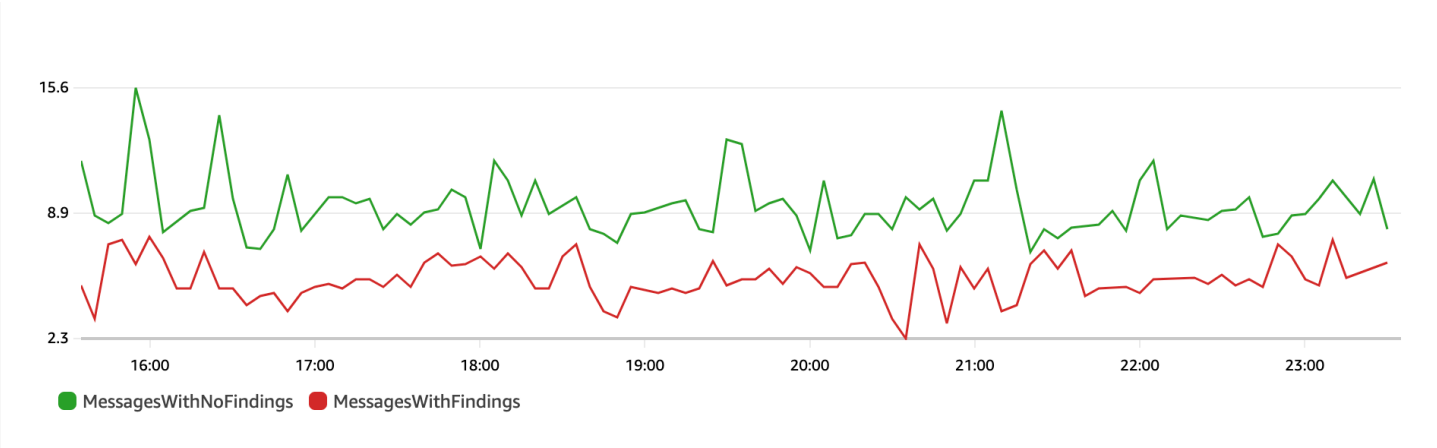
在以下示例中，`callerPrincipal`用于标识敏感内容的来源，并用作`messageID`对照`Publish`API 响应进行检查的参考。

```
{
  "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
  "auditTimestamp": "2022-05-12T2:10:44Z",
  "callerPrincipal": "arn:aws:iam::123412341234:role/Publisher",
  "resourceArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
  "dataIdentifiers": [
    {
      "name": "Name",
      "count": 1,
      "detections": [
        {
          "start": 1,
          "end": 2
        }
      ]
    },
    {
      "name": "PhoneNumber",
      "count": 2,
      "detections": [
        {
          "start": 3,
          "end": 4
        },
        {
          "start": 5,
```

```
            "end": 6
          }
        ]
      }
    ]
}
```

审计操作指标

当审计操作指定FindingsDestination或NoFindingsDestination属性时，主题所有者还会收到CloudWatchMessagesWithFindings和MessagesWithNoFindings指标。



去身份识别操作

去身份识别操作会遮蔽或去除所发布或已送达消息中的敏感数据。此操作既适用于入站消息，又适用于出站消息，需要以下类型的配置之一：

- MaskConfig— 使用下表中支持的字符进行掩码。例如，ssn: 123-45-6789 变成 ssn: ###########。

```
{
"Operation": {
    "Deidentify": {
        "MaskConfig": {
            "MaskWithCharacter": "#"
          }
    }
}
```

支持的遮蔽字符	名称
*	星号
A-Z、a-z 和 0-9	字母数字
	空格
!	感叹号
$	美元符号
%	百分号
&	& 符号
()	括号
+	加号
,	逗号
-	连字符
.	周期
/\	斜杠、反斜杠
#	数字符号
:	冒号
;	分号
=, <>	等于、小于或大于号
@	at 符号
[]	方括号
^	插入符号

支持的遮蔽字符	名称
_	下划线
`	反引号
|	竖线
~	波浪符号

- RedactConfig— 通过完全删除数据来进行编辑。例如，ssn: 123-45-6789 变成 ssn: 。

```
{
"Operation": {
    "Deidentify": {
        "RedactConfig": {}
    }
}
```

在入站消息上，在审计操作完成后，敏感数据会被取消标识，当整封消息都敏感时，SNS:PublishAPI调用者会收到以下无效参数错误。

```
Error code: AuthorizationError ...
```

拒绝操作

如果消息包含敏感数据，则拒绝操作会中断消息的PublishAPI请求或传送。拒绝操作对象为空，因为它不需要额外配置。

```
"Operation": {
    "Deny": {}
}
```

在入站消息中，来SNS:PublishAPI电者会收到授权错误。

```
Error code: AuthorizationError ...
```

在出站消息中，Amazon SNS 主题不会将消息传送给订阅。要跟踪未经授权的传输，请启用主题的传输状态日志记录。下面是传输状态日志示例：

```
{
```

```
    "notification": {
        "messageMD5Sum": "29638742ffb68b32cf56f42a79bcf16b",
        "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
        "topicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
        "timestamp": "2022-05-12T2:12:44Z"
    },
    "delivery": {
        "deliveryId": "98236591c-56aa-51ee-a5ed-0c7d43493170",
        "destination": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
        "providerResponse": "The topic's data protection policy prohibits this message
 from being delivered to <subscription-arn>",
        "dwellTimeMs":20,
        "attempts":1,
        "statusCode": 403
    },
    "status": "FAILURE"
}
```

亚马逊SNS数据保护政策示例

以下数据保护策略示例可用于审计和拒绝敏感数据。有关包含示例应用程序的完整教程，请参阅亚马逊消息数据保护简介SNS博客文章。

主题

- 审计策略示例
- 带有入站去身份识别遮蔽语句的策略示例
- 带有入站去身份识别去除语句的策略示例
- 带有出站去身份识别遮蔽语句的策略示例
- 带有入站去身份识别去除语句的策略示例
- 入站拒绝语句示例策略
- 出站拒绝语句示例策略

审计策略示例

审计策略允许您审计多达 99% 的入站邮件，并将调查结果发送到亚马逊 CloudWatch、Amazon Data Firehose 和亚马逊 S3。

例如，您可以创建审计策略，评估任何系统是否无意中发送或接收了敏感数据。如果您的审计结果表明系统将信用卡信息发送到了无需这些信息的系统，则可以实施数据保护策略来阻止传输此类数据。

以下示例通过查找信用卡号并将发现结果发送到 Lo CloudWatch gs、Firehose 和 Amazon S3 来审核通过该主题的 99% 的消息。

数据保护策略：

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": ["*"],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Audit": {
          "SampleRate": "99",
          "FindingsDestination": {
            "CloudWatchLogs": {
              "LogGroup": "<example log name>"
            },
            "Firehose": {
              "DeliveryStream": "<example stream name>"
            },
            "S3": {
              "Bucket": "<example bucket name>"
            }
          }
        }
      }
    }
  ]
}
```

审计结果格式示例：

```
{
    "messageId": "...",
    "callerPrincipal": "arn:aws:sts::123456789012:assumed-role/ExampleRole",
    "resourceArn": "arn:aws:sns:us-east-1:123456789012:ExampleArn",
    "dataIdentifiers": [
```

```
        {
            "name": "CreditCardNumber",
            "count": 1,
            "detections": [
                { "start": 1, "end": 2 }
            ]
        }
    ],
    "timestamp": "2021-04-20T00:33:40.241Z"
}
```

带有入站去身份识别遮蔽语句的策略示例

以下示例通过遮蔽消息内容中的敏感数据，阻止用户将带有 CreditCardNumber 的敏感消息发布到主题。

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "#"
          }
        }
      }
    }
  ]
}
```

入站去身份识别遮蔽结果示例：

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ################
```

带有入站去身份识别去除语句的策略示例

以下示例通过去除消息内容中的敏感数据，阻止用户将带有 CreditCardNumber 的敏感消息发布到主题。

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
        }
      }
    }
  ]
}
```

入站去身份识别去除结果示例：

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

带有出站去身份识别遮蔽语句的策略示例

以下示例通过遮蔽消息内容中的敏感数据，阻止用户使用 CreditCardNumber 接收消息。

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "-"
          }
        }
      }
    }
  ]
}
```

出站去身份识别遮蔽结果示例：

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ----------------
```

带有入站去身份识别去除语句的策略示例

以下示例通过去除消息内容中的敏感数据，阻止用户使用 CreditCardNumber 接收消息。

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
```

```
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
        }
      }
    }
  ]
}
```

出站去身份识别去除结果示例：

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

入站拒绝语句示例策略

以下示例阻止用户将消息内容中带有 `CreditCardNumber` 的消息发布到主题。API响应中被拒绝的负载的状态码为 “403 AuthorizationError”。

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
```

```
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deny": {}
      }
    }
  ]
}
```

出站拒绝语句示例策略

以下示例阻止 AWS 账户接收包含以下内容的消息`CreditCardNumber`。

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deny": {}
      }
    }
  ]
}
```

出站拒绝结果示例，已登录 Amazon CloudWatch：

```
{
  "notification": {
    "messageMD5Sum": "2e8f58ff2eeed723b56b15493fbfb5a5",
    "messageId": "8747a956-ebf1-59da-b291-f2c2e4b87c9c",
    "topicArn": "arn:aws:sns:us-east-2:664555388960:test1",
    "timestamp": "2022-09-08 15:40:57.144"
  },
  "delivery": {
```

```
    "deliveryId": "6a422437-78cc-5171-ad64-7fa3778507aa",
    "destination": "arn:aws:sqs:us-east-2:664555388960:test",
    "providerResponse": "The topic's data protection policy prohibits this message from
 being delivered to <subscription arn>",
    "dwellTimeMs": 22,
    "attempts": 1,
    "statusCode": 403
  },
  "status": "FAILURE"
}
```

在 Amazon 中创建数据保护政策 SNS

数据保护策略通过审计、取消身份识别（屏蔽或编辑）和拒绝（屏蔽）在应用程序或之间移动的敏感信息，帮助您保护发布到您的 Amazon SNS 主题的数据。 AWS 服务您可以使用 AWS API、 AWS CLI AWS CloudFormation、或 AWS Management Console 在 Amazon 中创建数据保护策略SNS。每个 Amazon SNS 主题只能定义一项政策。每个数据保护策略可以有一个或多个去身份识别和拒绝语句，但只能有一个审计语句。

主题

- SNS使用在 Amazon 中创建数据保护政策 API
- SNS使用 Amazon 创建数据保护政策 CLI
- SNS使用在 Amazon 中创建数据保护政策 CloudFormation
- SNS使用控制台在 Amazon 中创建数据保护策略
- SNS使用 Amazon 创建数据保护政策 SDK

SNS使用在 Amazon 中创建数据保护政策 API

一个 AWS 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service 端点和配额。

使用创建数据保护策略 API

您可以使用创建 Amazon SNS 数据保护政策 AWS API。

将数据保护政策与 Amazon SNS 主题一起创建 (AWS API)

使用标准 Amazon SNS 主题的`DataProtectionPolicy`属性：

- `CreateTopic`

为现有 Amazon SNS 主题检索或创建数据保护政策 (AWS API)

调用下列一项操作：

- GetDataProtectionPolicy
- PutDataProtectionPolicy

SNS使用 Amazon 创建数据保护政策 CLI

AWS 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service 端点和配额。

创建数据保护策略 (AWS CLI)

您可以使用创建 Amazon SNS 数据保护政策 AWS Command Line Interface。

将数据保护政策与 Amazon SNS 主题一起创建 (AWS CLI)

使用此选项创建新的数据保护策略以及标准的 Amazon SNS 主题：

- create-topic

为现有 Amazon SNS 主题创建或检索数据保护政策 (AWS CLI)

调用下列一项操作：

- get-data-protection-policy
- put-data-protection-policy

SNS使用在 Amazon 中创建数据保护政策 CloudFormation

AWS 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service 端点和配额。

创建数据保护策略 (CloudFormation)

您可以使用创建亚马逊SNS数据保护政策 AWS CloudFormation。

将数据保护政策与 Amazon SNS 主题一起创建 (CloudFormation)

使用此选项创建新的数据保护策略以及标准的 Amazon SNS 主题：

- AWS:::SNS: 话题

SNS使用控制台在 Amazon 中创建数据保护策略

AWS 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service 端点和配额。

创建数据保护策略和 Amazon SNS 主题（控制台）

使用此选项创建新的数据保护策略以及标准的 Amazon SNS 主题。

1. 登录 Amazon SNS 控制台。
2. 选择一个主题或创建一个新主题。有关创建主题的更多详情，请参阅创建 Amazon SNS 主题。
3. 在 Create topic（创建主题）页面的 Details（详细信息）部分，选择 Standard（标准）。
   a. 输入主题的名称。
   b. （可选）输入主题的显示名称。
4. 展开 Data protection policy（数据保护策略）。
5. 选择一个 Configuration mode（配置模式）：
   - Basic（基本）– 使用简单菜单定义数据保护策略。
   - 高级-使用定义自定义数据保护策略JSON。
6. （可选）要创建您自己的自定义数据标识符，请展开自定义数据标识符配置部分，执行以下操作：
   a. 为自定义数据标识符输入唯一名称。自定义数据标识符名称支持字母数字、下划线（_）和连字符（-）等字符。最多支持 128 个字符。此名称不能与托管式数据标识符同名。有关自定义数据标识符限制的完整列表，请参阅自定义数据标识符限制。
   b. 输入自定义数据标识符的正则表达式 (RegEx)。 RegEx支持字母数字字符、 RegEx 保留字符和符号。 RegEx 最大长度为 200 个字符。如果太复杂，Amazon SNS 将无法进行API通话。RegEx 有关 RegEx限制的完整列表，请参阅自定义数据标识符限制。
   c. （可选）选择添加自定义数据标识符以根据需要添加其它数据标识符。每项数据保护策略最多支持 10 个自定义数据标识符。
7. 选择您要添加到数据保护策略中的语句。您可以将审计、去身份识别（遮蔽或去除）和拒绝（阻止）语句类型添加到同一数据保护策略中。

a. Add audit statement（添加审计语句）– 配置要审计的敏感数据、要为该数据审计的消息百分比以及将审计日志发送到何处。

> **Note**
>
> 每个数据保护策略或主题仅允许使用一个审计语句。

i. 选择 data identifiers（数据标识符）以定义要审计的敏感数据。

ii. 对于 Audit sample rate（审计采样率），输入要在其中审计敏感信息的消息百分比，最大值为 99%。

iii. 在“审计目标”中，选择 AWS 服务 要发送审计结果的目的地，然后为您 AWS 服务 使用的每个目标输入目标名称。您可以从以下 Amazon Web Services 中进行选择：

- Amazon CloudWatch — CloudWatch Logs 是 AWS 标准的日志解决方案。使用 CloudWatch 日志，您可以使用 Logs Insights（参见此处的示例）执行日志分析，并创建指标和警报。 CloudWatch 日志是许多服务发布日志的地方，这使得使用一个解决方案可以更轻松地聚合所有日志。有关亚马逊的信息 CloudWatch，请参阅亚马逊 CloudWatch 用户指南。
- Amazon Data Firehos e — Firehose 可以满足实时直播到 Splunk 的需求，而 OpenSearch亚马逊 Redshift 可以满足进一步日志分析的需求。有关亚马逊数据 Firehose 的信息，请参阅亚马逊数据 Firehose 用户指南。
- Amazon Simple Storage Service – Amazon S3 是经济实惠的日志目标，可用于存档目的。您可能需要将日志保留数年。在这种情况下，您可以将日志放入 Amazon S3 中以节省成本。有关 Amazon Simple Storage Service 的信息，请参阅 Amazon Simple Storage Service 用户指南。

b. Add a de-identify statement（添加去身份识别语句）– 配置要在消息中去身份识别的敏感数据（无论是遮蔽还是去除该数据），并且账户将停止传输该数据。

i. 对于 Data Identifiers（数据标识符），选择要去身份识别的敏感数据。

ii. 在“为其定义此去身份化对账单”中，选择此去身份化声明适用的 AWS 账户或IAM委托人。您可以将其应用于所有 AWS 账户，也可以应用于使用 AWS 账户或IAM实体的特定账户或实IAM体（账户根、角色IDs或用户）ARNs。分隔多个IDs或ARNs使用逗号(,)。

支持以下IAM主体：

- IAM账户委托人-例如，`arn:aws:iam::AWS-account-ID:root`。
- IAM角色主体-例如，`arn:aws:iam::AWS-account-ID:role/role-name`。
- IAM用户主体-例如，`arn:aws:iam::AWS-account-ID:user/user-name`。

iii. 对于 De-identify Option（去身份识别选项），请选择要如何对敏感数据去身份识别。支持以下选项：

- Redact（去除）– 完全删除数据。例如，电子邮件 `classified@amazon.com` 将变为电子邮件 。
- Mask（遮蔽）– 使用单个字符替换数据。例如，电子邮件 `classified@amazon.com` 将变为电子邮件 `**********`。

iv. （可选）根据需要继续添加去身份识别语句。

c. Add deny statement（添加拒绝语句）– 配置要在您的主题中阻止传输哪些敏感数据，以及阻止哪些主体传输这些数据。

i. 对于 Data Direction（数据方向），请选择拒绝语句的消息方向：

- Inbound messages（入站消息）– 将此拒绝语句应用于发送到该主题的消息。
- Outbound messages（出站消息）– 将此拒绝语句应用于主题传输到订阅端点的消息。

ii. 选择 Data Identifiers（数据标识符）以定义要拒绝的敏感数据。

iii. 选择适用于IAM此拒绝语句的主体。您可以将其应用于所有 AWS 账户 AWS 账户、特定账户或使用账户或IAM实体的实体（例如，账户根、角色IDs或IAM用户）ARNs。分隔多个IDs或ARNs使用逗号 (,)。支持以下IAM主体：

- IAM账户委托人-例如，`arn:aws:iam::AWS-account-ID:root`。
- IAM角色主体-例如，`arn:aws:iam::AWS-account-ID:role/role-name`。
- IAM用户主体-例如，`arn:aws:iam::AWS-account-ID:user/user-name`。

iv. （可选）根据需要继续添加拒绝语句。

SNS使用 Amazon 创建数据保护政策 SDK

一个 AWS 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service 端点和配额。

创建数据保护策略 (AWS SDK)

您可以使用创建 Amazon SNS 数据保护政策 AWS SDK。

将数据保护政策与 Amazon SNS 主题一起创建 (AWS SDK)

使用以下选项创建新的数据保护策略以及标准的 Amazon SNS 主题：

Java

```
/**
 * For information regarding CreateTopic see this documentation topic:
 *
 * https://docs.aws.amazon.com/code-samples/latest/catalog/javav2-sns-src-main-java-
com-example-sns-CreateTopic.java.html
 */

public static String createSNSTopicWithDataProtectionPolicy(SnsClient snsClient,
 String topicName, String dataProtectionPolicy) {

    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .dataProtectionPolicy(dataProtectionPolicy)
                .build();

        CreateTopicResponse result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

JavaScript

```
// Import required AWS SDK clients and commands for Node.js
import {CreateTopicCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = { Name: "TOPIC_NAME", DataProtectionPolicy:
 "DATA_PROTECTION_POLICY" };
```

```
const run = async () => {
  try {
    const data = await snsClient.send(new CreateTopicCommand(params));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

为现有 Amazon SNS 主题创建或检索数据保护政策 (AWS SDK)

使用以下选项创建或检索新的数据保护策略以及标准的 Amazon SNS 主题：

Java

```
public static void putDataProtectionPolicy(SnsClient snsClient, String topicName,
 String dataProtectionPolicy) {

    try {
        PutDataProtectionPolicyRequest request =
 PutDataProtectionPolicyRequest.builder()
                .resourceArn(topicName)
                .dataProtectionPolicy(dataProtectionPolicy)
                .build();

        PutDataProtectionPolicyResponse result =
 snsClient.putDataProtectionPolicy(request);
        System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                + "\n\nTopic " + request.resourceArn()
                + " DataProtectionPolicy " + request.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}


public static void getDataProtectionPolicy(SnsClient snsClient, String topicName) {
```

```
    try {
        GetDataProtectionPolicyRequest request =
 GetDataProtectionPolicyRequest.builder()
                .resourceArn(topicName)
                .build();

        GetDataProtectionPolicyResponse result =
 snsClient.getDataProtectionPolicy(request);

        System.out.println("\n\nStatus is " + result.sdkHttpResponse().statusCode()
        + "\n\nDataProtectionPolicy: \n\n" + result.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

JavaScript

```
// Import required AWS SDK clients and commands for Node.js
import {PutDataProtectionPolicyCommand, GetDataProtectionPolicyCommand } from "@aws-
sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const putParams = { ResourceArn: "TOPIC_ARN", DataProtectionPolicy:
 "DATA_PROTECTION_POLICY" };

const runPut = async () => {
  try {
    const data = await snsClient.send(new
 PutDataProtectionPolicyCommand(putParams));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
runPut();

// Set the parameters
const getParams = { ResourceArn: "TOPIC_ARN" };
```

```
const runGet = async () => {
  try {
    const data = await snsClient.send(new
 GetDataProtectionPolicyCommand(getParams));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
runGet();
```

删除 Amazon 中的数据保护政策 SNS

您可以使用 AWS API、、 AWS CLI或删除 Amazon SNS 数据保护政策 AWS Management Console。 AWS CloudFormation

有关 Amazon SNS 数据保护政策的一般信息，请参阅了解亚马逊的数据保护政策 SNS。

AWS 账户中 Amazon SNS 数据保护政策资源的数量和大小是有限的。有关更多信息，请参阅中的 Amazon SNS API 限制。 AWS 一般参考

主题

- 使用控制台删除数据保护策略
- 使用空JSON字符串删除数据保护策略
- 使用删除数据保护策略 AWS CLI

使用控制台删除数据保护策略

使用控制台删除托管数据保护策略

1. 登录 Amazon SNS 控制台。
2. 选择包含您要删除的数据保护策略的主题。
3. 选择编辑。
4. 展开 Data protection policy（数据保护策略）部分。
5. 在您要删除的数据保护策略语句旁边，选择 Remove（删除）。
6. 选择 Save changes（保存更改）。

使用空JSON字符串删除数据保护策略

您可以通过将数据保护策略更新为空JSON字符串来删除该策略。

使用删除数据保护策略 AWS CLI

您可以使用 AWS CLI删除数据保护策略。

```
//aws sns put-data-protection-policy --resource-arn topic-arn --data-
protection-policy ""
```

Amazon SNS 数据标识符

Amazon SNS 使用多种标准和技术（包括机器学习和模式匹配）来检测敏感数据。这些标准和技术统称为数据标识符，可以检测许多国家/地区和区域的大量且不断增长的敏感数据类型。Amazon SNS 托管数据标识符提供预配置的数据类型，用于保护财务数据、个人健康信息 (PHI) 和个人身份信息 (PII)。您还可以使用自定义数据标识符来创建您自己的针对您的特定用例量身定制的数据标识符。

主题

- 在 Amazon 中使用托管数据标识符 SNS
- 在 Amazon 中使用自定义数据标识符 SNS

在 Amazon 中使用托管数据标识符 SNS

主题

- 什么是托管数据标识符？
- Amazon SNS 敏感数据类型：凭证
- Amazon SNS 敏感数据类型：设备
- Amazon SNS 敏感数据类型：财务
- Amazon SNS 敏感数据类型：受保护的健康信息 (PHI)
- Amazon SNS 敏感数据类型：个人身份信息 (PII)

什么是托管数据标识符？

Amazon SNS 托管数据标识符旨在检测特定类型的敏感数据，例如信用卡号、 AWS 秘密访问密钥或特定国家或地区的护照号码。在创建数据保护策略时，您可以将 Amazon 配置SNS为使用这些标识符来分析通过该主题的消息，并在检测到消息时采取措施。

Amazon SNS 可以使用托管数据标识符检测以下类别的敏感数据：

- 证书，例如私钥或私有访问 AWS 密钥
- 设备标识符，例如 IP 地址或MAC地址
- 财务信息，例如信用卡号
- 健康信息，PHI例如健康保险或医疗身份证号码
- 个人信息，PII例如驾照或社会安全号码

在每个类别中，Amazon SNS 可以检测多种类型的敏感数据。本节中的主题列出并描述了各种类型以及对其进行检测的相关要求。对于每种类型，它们还说明了托管数据标识符的唯一标识符 (ID)，此标识符设计用于检测数据。创建数据保护策略时，您可以使用此 ID 包含托管数据标识符，供消息数据保护功能进行检测。

关键字要求

为了检测某些类型的敏感数据，Amazon SNS 会扫描数据附近的关键字。如果特定类型的数据属于这种情况，此部分中的后续主题将说明该数据的特定关键字要求。

关键字不区分大小写。此外，如果关键词包含空格，Amazon SNS 会自动匹配不包含空格、不包含下划线 (_) 或连字符 (-) 而不是空格的关键词变体。在某些情况下，亚马逊SNS还会扩展或缩写关键词以解决关键词的常见变体。

Amazon SNS 托管敏感数据类型的数据标识符

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测到的凭证、设备、财务、医疗和个人健康信息 (PHI) 的类型。除此之外，某些类型的数据也可能被视为个人身份信息 (PII)。

区域相关的数据标识符要求标识符名称带破折号，以及两个字母 (ISO3166-1 alpha-2) 代码。例如，DriversLicense-US。

标识符	类别	国家/地区/语言
BankAccountNumber	财务	DE、ES、FR、GB、IT
CepCode	个人	BR
Cnpj	个人	BR
CpfCode	个人	BR

标识符	类别	国家/地区/语言
DriversLicense	个人	AT、AU、BE、 BG、CA、CY、 CZ、DE、DK、EE、ES、FI、 FR、GB、GR、 HR、HU、IE、IT、LT、LU、 LV、MT、NL、 PL、PT、RO、SE、SI、SK、 US
DrugEnforcementAgencyNumber	健康	美国
ElectoralRollNumber	个人	GB
HealthInsuranceCardNumber	健康	EU
HealthInsuranceClaimNumber	健康	美国
HealthInsuranceNumber	健康	FR
HealthcareProcedureCode	健康	美国
IndividualTaxIdentificationNumber	个人	美国
InseeCode	个人	FR
MedicareBeneficiaryNumber	健康	美国
NationalDrugCode	健康	美国
NationalIdentificationNumber	个人	DE、ES、IT
NationalInsuranceNumber	个人	GB
NationalProviderId	健康	美国
NhsNumber	健康	GB

标识符	类别	国家/地区/语言
NieNumber	个人	ES
NifNumber	个人	ES
PassportNumber	个人	CA、DE、ES、 FR、GB、IT、US
PermanentResidenceNumber	个人	CA
PersonalHealthNumber	健康	CA
PhoneNumber	个人	BR、DE、ES、 FR、GB、IT、US
PostalCode	个人	CA
RgNumber	个人	BR
SocialInsuranceNumber	个人	CA
Ssn	个人	ES、US
TaxId	个人	DE、ES、FR、GB
ZipCode	个人	美国

与语言/地区无关的受支持的标识符

标识符	类别
地址	个人
AwsSecretKey	凭证
CreditCardExpiration	财务
CreditCardNumber	财务

标识符	类别
CreditCardSecurityCode	财务
EmailAddress	个人
IpAddress	个人
LatLong	个人
名称	个人
OpenSshPrivateKey	凭证
PgpPrivateKey	凭证
PkcsPrivateKey	凭证
PuttyPrivateKey	凭证
VehicleIdentificationNumber	个人

Amazon SNS 敏感数据类型：凭证

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测到的凭证类型。

检测类型	托管数据标识符 ID	所需关键字	国家和地区
AWS 秘密访问密钥	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	任何
打开SSH私钥	OpenSshPrivateKey	否	任何
PGP私钥	PgpPrivateKey	否	任何
公钥加密标准 (PKCS) 私钥	PkcsPrivateKey	否	任何

检测类型	托管数据标识符 ID	所需关键字	国家和地区
Pu TTY 私钥	PuttyPrivateKey	否	任何

凭证数据ARNs类型的数据标识符

以下列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARNs)。

凭证数据标识符 ARNs
arn: aws: dataprotection:: aws: 数据标识符/ AwsSecretKey
arn: aws: dataprotection:: aws: 数据标识符/ OpenSshPrivateKey
arn: aws: dataprotection:: aws: 数据标识符/ PgpPrivateKey
arn: aws: dataprotection:: aws: 数据标识符/ PkcsPrivateKey
arn: aws: dataprotection:: aws: 数据标识符/ PuttyPrivateKey

Amazon SNS 敏感数据类型：设备

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测到的设备标识符类型。

检测类型	托管数据标识符 ID	所需关键字	国家和地区
IP 地址	IpAddress	否	任何

设备数据类型的数据ARNs标识符

以下列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARNs)。

设备数据标识符 ARN
arn: aws: dataprotection:: aws: 数据标识符/ IpAddress

Amazon SNS 敏感数据类型：财务

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测到的财务信息类型。

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
银行账号	BankAccountNumber BankAccountNumber-US	是，请参阅 银行账号的关键字。	这包括：由最多 34 个字母数字字符组成的国际银行账号 (IBANs)，包括国家代码等元素。	法国、德国、意大利、西班牙、英国
信用卡到期日期	CreditCardExpiration	exp d、exp m、exp y、expiration、expiry	–	任何
信用卡磁条数据	CreditCardMagneticStripe	是，包括：card data、iso7813、mag、magstripe、stripe、swipe。	这包括轨道 1 和 2。	任何
信用卡号	CreditCardNumber	account number、american express、amex、bank card、card 、card num、card number、cc #、ccn、check card、credit、credit card#、dan	检测要求数据为符合卢恩支票公式的13-19位数序列，并对以下任何类型的信用卡使用标准卡号前缀：美国运通、Dankort、Diner's Club、Discover、Electron、日本信用卡局 (JCB)	任何

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
		kort、debit、debit card、diners club、discover、electron、elo verification code、japanese card bureau、jcb、mastercard、mc、pan、payment account number、payment card number、pcn、unionpay、visa	UnionPay、万事达卡和Visa（下方的上标链接 1）。	

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
信用卡验证码	CreditCardSecurityCode	card id、card identification code、card identification number、card security code、card validation code、card validation number、card verification data、card verification value、cvc、cvc2、cvv、cvv2、elo verification code	–	任何

1. Amazon SNS 不会报告出现以下序列的情况，信用卡发卡机构已保留这些顺序供公开测试：

   122000000000003、2222405343248877、2222990905257051、2223007648726984、2223577120017 6 和 76009244561。

银行账号的关键字

使用以下关键字检测由最多 34 个字母数字字符组成的国际银行账号 (IBANs)，包括国家代码等元素。

国家或地区	关键字			
法国	account code, account number, accountno			

国家或地区	关键字			
	#, accountnumber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte			
德国	account code, account number, accountno #, accountnumber#, bankleitzahl, bban, customer account id, customer account number, customer bank account id, geheimzahl, iban, kartennummer, kontonummer, kreditkartennummer, sepa			

国家或地区	关键字			
意大利	account code, account number, accountno #, accountnu mber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			
西班牙	account code, account number, accountno #, accountnu mber#, bban, código cuenta, código cuenta bancaria, cuenta cliente id, customer account ID, customer account number, customer bank account id, iban, número cuenta bancaria cliente, número cuenta cliente			

国家或地区	关键字			
UK	account code, account number, accountno #, accountnu mber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			
美国	bank account、b ank acct、chec king account、c hecking acct、depo sit account、d eposit acct、savi ngs account、s avings acct、cheq uing account、c hequing acct			

财务数据类型的数据ARNs标识符

以下列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARNs)。

财务数据标识符 ARNs
arn: aws: 数据保护:: aws: data-identifier/-DE BankAccountNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES BankAccountNumber

财务数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identifir BankAccountNumber
arn: aws: dataprotection:: aws: data-identifer BankAccountNumber
arn: aws: dataprotection:: aws: data-it BankAccountNumber
arn: aws: dataprotection:: aws: data-identif BankAccountNumber
arn: aws: dataprotection:: aws: 数据标识符/ CreditCardExpiration
arn: aws: dataprotection:: aws: 数据标识符/ CreditCardNumber
arn: aws: dataprotection:: aws: 数据标识符/ CreditCardSecurityCode

Amazon SNS 敏感数据类型：受保护的健康信息 (PHI)

下表列出并描述了 Amazon SNS 可以使用托管数据标识符检测到的受保护健康信息 (PHI) 的类型。

检测类型	托管数据标识符 ID	所需关键字	国家和地区
缉毒机构 (DEA) 注册号	DrugEnforcementAgencyNumber	dea number, dea registration	美国
Health Insurance 卡号 (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero、carta assicurazione numero、carte d’assurance maladie、carte européenne d'assurance maladie、ceam、ehic、ehic#、finlandehicnumber#、gesundheitskarte、hälsokort、health card、health	EU

检测类型	托管数据标识符 ID	所需关键字	国家和地区
		card number、health insurance card、health insurance number、insurance card number、krankenversicherungskarte、krankenversicherungsnummer、medical account number、numero conto medico、numéro d’assurance maladie、numéro de carte d’assurance、numéro de compte medical、número de cuenta médica、número de seguro de salud、número de tarjeta de seguro、sairaanhoitokortin、sairausvakuutuskortti、sairausvakuutus numero、sjukförsäkring nummer、sjukförsäkringskort、suomi ehic-numero、tarjeta de salud、terveyskortti、tessera sanitaria assicurazione	

检测类型	托管数据标识符 ID	所需关键字	国家和地区
		numero、versicherungsnummer	
Health Insurance 索赔编号 (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	美国
健康保险或医疗识别号	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR
医疗保健通用程序编码系统 (HCPCS) 代码	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	美国
医疗保险受益人号码 () MBN	MedicareBeneficiaryNumber	mbi, medicare beneficiary	美国
《国家药品法》(NDC)	NationalDrugCode	national drug code, ndc	美国
国家提供商标识符 (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	美国
国家卫生服务 (NHS) 号码	NhsNumber	national health service, NHS	GB
个人健康号码 (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

健康保险和医疗识别号的关键字

为了检测各种类型的健康保险和医疗识别码，Amazon SNS 要求关键词必须接近这些数字。这包括欧洲健康保险卡号（欧盟、芬兰）、健康保险号码（法国）、医疗保险受益人标识符（美国）、国民保险号码（英国）、NHS号码（英国）和个人健康号码（加拿大）。

下表列出了亚马逊SNS识别的特定国家和地区的关键词。

国家或地区	关键词
加拿大	Canada healthcare number, msp number, personal healthcare number, phn, soins de santé
EU	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d’assurance maladie, numéro de carte d’assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer
芬兰	ehic, ehic#, finland health insurance card, finlandehicnumber#, finska sjukförsäkringskort, hälsokort, health card, health card number, health insurance card, health insurance

国家或地区	关键词
	number, sairaanhoitokortin, sairaanhoitokortin , sairausvakuutuskortti, sairausvakuutusnum ero, sjukförsäkring nummer, sjukförsäkringskor t, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti
法国	carte d'assuré social, carte vitale, insurance card
英国	国家卫生服务 NHS
美国	mbi, medicare beneficiary

受保护ARNs的健康信息数据类型的数据标识符 (PHI)

下面列出了可在数据保护策略中PHI使用的数据标识符 Amazon 资源名称 (ARNs)。

PHI数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identif DrugEnforcementAgencyNumber
arn: aws: dataprotection:: aws: data-identif HealthcareProcedureCode
arn: aws: dataprotection:: aws: data-identifer HealthInsuranceCardNumber
arn: aws: dataprotection:: aws: data-identif HealthInsuranceClaimNumber
arn: aws: dataprotection:: aws: data-identifir HealthInsuranceNumber
arn: aws: dataprotection:: aws: data-identif MedicareBeneficiaryNumber
arn: aws: dataprotection:: aws: data-identif NationalDrugCode
arn: aws: dataprotection:: aws: data-identifer NationalInsuranceNumber
arn: aws: dataprotection:: aws: data-identif NationalProviderId
arn: aws: dataprotection:: aws: data-identifer NhsNumber

PHI数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identifir PersonalHealthNumber

Amazon SNS 敏感数据类型：个人身份信息 (PII)

下表列出并描述了 Amazon 使用托管数据标识符SNS可以检测到的个人身份信息类型 (PII)。

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
出生日期	DateOfBirth	dob, date of birth, birthdate , birth date, birthday, b-day, bday	支持包括大多数日期格式，例如所有数字以及数字和月份名称的组合。日期组件可以用空格、斜杠（/）或连字符（-）分隔。	任何
邮政编码 () CEP	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	巴西
国家法人地籍 () CNPJ	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	巴西
物理人员地籍 () CPF	CpfCode	Cadastro de pessoas fisicas, cadastro de	–	巴西

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
		pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf		
驾驶执照识别号	DriversLicense	是，请参阅 驾驶执照识别号的关键字。	–	澳大利亚、奥地利、比利时、保加利亚、加拿大、克罗地亚、塞浦路斯、捷克共和国、丹麦、爱沙尼亚、芬兰、法国、德国、希腊、匈牙利、爱尔兰、意大利、拉脱维亚、立陶宛、卢森堡、马耳他、荷兰、波兰、葡萄牙、罗马尼亚、斯洛伐克、斯洛文尼亚、西班牙、瑞典、英国、美国

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
选民名册编号	ElectoralRollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	UK
个人纳税人识别号	IndividualTaxIdentificationNumber	是，请参阅 纳税人识别号和参考号的关键字。	–	美国
国家统计与经济研究所 (INSEE)	InseeCode	是，请参阅 国民身份证号的关键字。	–	法国
身份证号码	NationalIdentificationNumber	是，请参阅 国民身份证号的关键字。	这包括 Documento Nacional de Identidad (DNI) 标识符（西班牙）、Codice fiscale 代码（意大利）和国民身份证号码（德语）。	德国、意大利、西班牙

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
国民保险号码 (NINO)	NationalI nsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationali nsurance# , nationali nsurancen umber, nin, nino	–	英国
外星人身份号码 () NIE	NieNumber	是，请参阅 纳税人识别号和参考号的关键字。	–	西班牙
税务识别号码 () NIF	NifNumber	是，请参阅 纳税人识别号和参考号的关键字。	–	西班牙
护照编号	PassportNumber	是，请参阅 护照号码的关键字。	–	加拿大、法国、德国、意大利、西班牙、英国、美国

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
永久居留号码	Permanent Residence Number	carte résident permanent , numéro carte résident permanent, numéro résident permanent , permanent resident card, permanent resident card number, permanent resident no, permanent resident no., permanent resident number, pr no, pr no., pr non, pr number, résident permanent no., résident permanent non	–	加拿大

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
电话号码	PhoneNumber	巴西：关键字还包括 cel、celular、fone、móvel、número residencial、numero residencial、telefone 其他：cell、contact、fax、fax number、mobile、phone、phone number、tel、telephone、telephone number	这包括美国的免费电话号码和传真号码。如果关键字靠近数据，则数字不必包含国家/地区代码。如果关键字并不靠近数据，则数字必须包含国家/地区代码。	巴西、加拿大、法国、德国、意大利、西班牙、英国、美国
邮政编码	PostalCode	No	–	加拿大
Registro Geral (RG)	RgNumber	是，请参阅 国民身份证号的关键字。	–	巴西
社会保险号码 (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	加拿大

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
社会安全号码 (SSN)	Ssn	西班牙 – número de la seguridad social、so cial security no.、social security no. número de la seguridad social、so cial security number、so cialsecur ityno#、ss n、ssn# 美国 – social security、 ss#、ssn	–	西班牙、美国
纳税人识别号或参考号	TaxId	是，请参阅 纳税人识别号和参考号的关键字。	这包括TIN（法国）；Steueri dentifika tionsnumm er（德国）；（西班牙）；以及CIF（英国）。TRN UTR	法国、德国、西班牙、英国
美国邮政编码	ZipCode	zip code, zip+4	–	美国

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
邮寄地址	Address	No	尽管不需要关键字，但检测要求地址中包含城市或地点的名称以及ZIP或邮政编码。	澳大利亚、加拿大、法国、德国、意大利、西班牙、英国、美国
电子邮件地址	EmailAddress	电子邮件、电子邮件地址、邮件、邮件地址	–	任何
全球定位系统 (GPS) 坐标	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	如果纬度和经度GPS坐标成对存储并且采用十进制 (DD) 格式，例如 41.948614、-87.655311，则亚马逊SNS可以检测坐标。Support 不包括度十进制分钟 (DDM) 格式的坐标，例如 41°56.9168'N 87°39.3187'W，或度、分、秒 () 格式，例如 41°56'55.0104" N 87°39'19.1196" W 87°39'19.1196" W。DMS	任何

检测类型	托管数据标识符 ID	所需关键字	其他信息	国家和地区
全名	Name	No	Amazon SNS 只能检测全名。只支持拉丁字符集。	任何
车辆识别码 (VIN)	VehicleIdentificationNumber	Fahrgestellnummer, niv, numarul de identificare, numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóviles, numéro d'identification du véhicule, vehicle identification number, vin, VIN numeris	亚马逊SNSVINs可以检测到由 17 个字符组成的序列并符合 ISO 3779 和 3780 标准。这些标准旨在供全球使用。	任何

驾驶执照识别号的关键字

为了检测各种类型的驾照识别码，Amazon SNS 要求关键词必须与这些号码相近。下表列出了亚马逊SNS识别的特定国家和地区的关键词。

国家或地区	关键词
澳大利亚	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license,

国家或地区	关键词
	drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
奥地利	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
比利时	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
保加利亚	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
加拿大	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit, permis de conduire
克罗地亚	vozačka dozvola
塞浦路斯	άδεια οδήγησης
捷克共和国	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz

国家或地区	关键词
丹麦	kørekort, kørekortnummer
爱沙尼亚	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
芬兰	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
法国	permis de conduire
德国	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
希腊	δεια οδήγησης, adeia odigisis
匈牙利	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
爱尔兰	ceadúnas tiomána
意大利	patente di guida, patente di guida numero, patente guida, patente guida numero
拉脱维亚	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
立陶宛	vairuotojo pažymėjimas
卢森堡	fahrerlaubnis, führerschäin
马耳他	liċenzja tas-sewqan
荷兰	permis de conduire, rijbewijs, rijbewijsnummer

国家或地区	关键词
波兰	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
葡萄牙	carta de condução, carteira de habilitação, carteira de motorist, carteira habilitação, carteira motorist, licença condução, licença de condução, número de licença, número licença, permissão condução, permissão de condução
罗马尼亚	numărul permisului de conducere, permis de conducere
斯洛伐克	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
斯洛文尼亚	vozniško dovoljenje
西班牙	carnet conducer, el carnet de conducer, licencia conducer, licencia de manejo, número carnet conducer, número de carnet de conducer, número de permiso conducer, número de permiso de conducer, número licencia conducer, número permiso conducer, permiso conducción, permiso conducer, permiso de conducción
瑞典	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

国家或地区	关键词
UK	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
美国	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

国民身份证号的关键字

为了检测各种类型的国民识别码，Amazon SNS 要求关键词必须与这些数字非常接近。这包括 Documento Nacional de Identidad (DNI) 标识符（西班牙）、法国国家统计和经济研究所（INSEE）代码、德国国民身份证号码和注册总局（RG）号码（巴西）。

下表列出了亚马逊SNS识别的特定国家和地区的关键词。

国家或地区	关键词
巴西	registro geral, rg
法国	assurance sociale, carte nationale d’identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité

国家或地区	关键词
	sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
德国	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
意大利	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
西班牙	dni, dni#, dninúmero#, documento nacional de identidad, identidad único, identidadúnico#, insurance number, national identification number, national identity, nationalid#, nationali dno#, número nacional identidad, personal identification number, personal identity no, unique identity number, uniqueid#

护照号码的关键字

为了检测各种类型的护照号码，Amazon SNS 要求关键词必须与护照号码相近。下表列出了亚马逊SNS识别的特定国家和地区的关键词。

国家或地区	关键词
加拿大	passeport, passeport#, passport, passport#, passportno, passportno#
法国	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non

国家或地区	关键词
德国	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
意大利	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
西班牙	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
UK	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
美国	passport, travel document

纳税人识别号和参考号的关键字

为了检测各种类型的纳税人识别号和参考号，Amazon SNS 要求关键词必须接近这些数字。下表列出了亚马逊SNS识别的特定国家和地区的关键词。

国家或地区	关键词
巴西	cadastro de pessoa física, cadastro de pessoa fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj, cpf
法国	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#

国家或地区	关键词
德国	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
西班牙	cif, cif número, cifnúmero#, nie, nif, número de contribuyente, número de identidad de extranjero, número de identificación fiscal, número de impuesto corporativo, personal tax number, tax id, tax identification number, tax number, tin, tin#
UK	paye, tax id, tax id no., tax id number, tax identification, tax identification#, tax no., tax number, tax reference, tax#, taxid#, temporary reference number, tin, trn, unique tax reference, unique taxpayer reference, utr
美国	个人纳税人识别号（itin, i.t.i.n.）

个人身份信息ARNs的数据标识符 (PII)

下表列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARNs)。

PII数据标识符 ARNs
arn:aws:dataprotection::aws:data-identifier/Address
arn: aws: dataprotection:: aws: data-identifer CepCode
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn: aws: dataprotection:: aws: data-identifer CpfCode
arn: aws: dataprotection:: aws: 数据标识符/ DateOfBirth
arn: aws: dataprotection:: aws: 数据标识符/-AT DriversLicense

PII数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: data-identifier/ DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-DE DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-DK DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-EE DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-ES DriversLicense
arn: aws: dataprotection:: aws: data-ident DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-HU DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-IE DriversLicense
arn: aws: dataprotection:: aws: data-it DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-LT DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-LU DriversLicense

PII数据标识符 ARNs
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: data-identif DriversLicense
arn: aws: dataprotection:: aws: data-identifier/ DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-PT DriversLicense
arn: aws: dataprotection:: aws: 数据标识符/-RO DriversLicense
arn: aws: 数据保护:: aws: data-identifier/-SE DriversLicense
arn: aws: dataprotection:: aws: data-identifer DriversLicense
arn: aws: dataprotection:: aws: data-identifir DriversLicense
arn: aws: dataprotection:: aws: data-identif DriversLicense
arn: aws: dataprotection:: aws: data-identifer ElectoralRollNumber
arn: aws: dataprotection:: aws: 数据标识符/ EmailAddress
arn: aws: dataprotection:: aws: data-identif IndividualTaxIdentificationNumber
arn: aws: dataprotection:: aws: data-identifir InseeCode
arn: aws: dataprotection:: aws: 数据标识符/ LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn: aws: 数据保护:: aws: data-identifier/-DE NationalIdentificationNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES NationalIdentificationNumber
arn: aws: dataprotection:: aws: data-it NationalIdentificationNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES NieNumber

PII数据标识符 ARNs
arn: aws: dataprotection:: aws: 数据标识符/-ES NifNumber
arn: aws: dataprotection:: aws: data-identifir PassportNumber
arn: aws: 数据保护:: aws: data-identifier/-DE PassportNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES PassportNumber
arn: aws: dataprotection:: aws: data-identifir PassportNumber
arn: aws: dataprotection:: aws: data-identifer PassportNumber
arn: aws: dataprotection:: aws: data-it PassportNumber
arn: aws: dataprotection:: aws: data-identif PassportNumber
arn: aws: dataprotection:: aws: data-identifir PermanentResidenceNumber
arn: aws: dataprotection:: aws: data-identifer PhoneNumber
arn: aws: 数据保护:: aws: data-identifier/-DE PhoneNumber
arn: aws: dataprotection:: aws: 数据标识符/-ES PhoneNumber
arn: aws: dataprotection:: aws: data-identifir PhoneNumber
arn: aws: dataprotection:: aws: data-identifer PhoneNumber
arn: aws: dataprotection:: aws: data-it PhoneNumber
arn: aws: dataprotection:: aws: data-identif PhoneNumber
arn: aws: dataprotection:: aws: data-identifir PostalCode
arn: aws: dataprotection:: aws: data-identifer RgNumber
arn: aws: dataprotection:: aws: data-identifir SocialInsuranceNumber
arn:aws:dataprotection::aws:data-identifier/Ssn-ES

PII数据标识符 ARNs
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn: aws: 数据保护:: aws: data-identifier/-DE TaxId
arn: aws: dataprotection:: aws: 数据标识符/-ES TaxId
arn: aws: dataprotection:: aws: data-identifir TaxId
arn: aws: dataprotection:: aws: data-identifer TaxId
arn: aws: dataprotection:: aws: 数据标识符/ VehicleIdentificationNumber
arn: aws: dataprotection:: aws: data-identif ZipCode

在 Amazon 中使用自定义数据标识符 SNS

主题

什么是自定义数据标识符？

自定义数据标识符 (CDIs) 允许您定义自己的自定义正则表达式，这些正则表达式可以在您的数据保护策略中使用。使用自定义数据标识符，您可以定位托管数据标识符无法提供的特定于业务的个人身份信息 (PII) 用例。例如，您可以使用自定义数据标识符来查找公司特定的员工。IDs自定义数据标识符可以与托管式数据标识符结合使用。

在数据保护策略中使用自定义数据标识符

以下数据保护政策指示 Amazon SNS 主题检测载有公司特定员工的有效负载IDs，然后IDs使用哈希符号 (#) 掩盖这些负载。

1. 在您的数据保护策略中创建一个 `Configuration` 块。
2. 为您的自定义数据标识符输入 `Name`。例如，**`EmployeeId`**。
3. 为您的自定义数据标识符输入 `Regex`。例如，**`EID-\d{9}-US`**。

4. 请参阅策略声明中的以下自定义数据标识符。

```
{
 "Name": "__example_data_protection_policy",
 "Description": "Example data protection policy",
 "Version": "2021-06-01",
 "Configuration": {
   "CustomDataIdentifier": [
     {"Name": "EmployeeId", "Regex": "EID-\d{9}-US"}
   ]
 },
 "Statement": [
   {
     "DataDirection": "Inbound",
     "Principal": ["*"],
     "DataIdentifier": [
        "EmployeeId"
     ],
     "Operation": {
       "Deidentify": {
         "MaskConfig": {
           "MaskWithCharacter": "#"
         }
       }
     }
   }
 ]
}
```

5. （可选）根据需要继续向 Configuration 块添加其他自定义数据标识符。数据保护策略目前支持最多 10 个自定义数据标识符。

自定义数据标识符限制

Amazon SNS 自定义数据标识符有以下限制：

- 每项数据保护策略最多支持 10 个自定义数据标识符。
- 自定义数据标识符名称最多可包含 128 个字符。支持以下字符：
  - 字母数字：(a-zA-Z0-9)
  - 符号：('_' | '-')
- RegEx 的最大长度为 200 个字符。支持以下字符：

  - 字母数字：(a-zA-Z0-9)
  - 符号：('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | ' ')
  - RegEx 保留字符：('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' | '\\' | '*' | '+' | '.')
- 自定义数据标识符不能与托管式数据标识符同名。
- 必须在每个 Amazon SNS 主题的数据保护政策中指定自定义数据标识符。

亚马逊SNS消息传送

本主题介绍 Amazon 如何SNS处理各种场景中的消息传送。您将了解原始消息传输，即Amazon SNS以未经修改的原始格式将消息传送到终端节点。您还将了解如何将来自亚马逊SNS主题的消息发送到另一个 Amazon SQS 队列 AWS 账户，从而深入了解跨账户消息传送。

本主题提供有关向不同的 Amazon SQS 队列或 Lambda 函数传送亚马逊SNS消息的情况 AWS 区域、跨区域传送的工作原理以及所涉及的注意事项的信息。

此外，你还将学习如何监控和解释邮件传送状态，从而提供有关邮件是否成功送达或遇到问题的关键信息。在消息传送失败的情况下，您将了解消息传送重试流程，包括 Amazon 如何SNS自动尝试重新传送消息以确保消息到达预期目的地。本主题还讨论了如何使用死信队列来捕获多次尝试后无法传送的消息，从而使您能够有效地分析和排除这些故障。

主题

Amazon SNS 原始消息传送

为了避免让 Amazon Data Firehose SQS、Amazon 和 HTTP/S 端点处理消息的JSON格式，亚马逊SNS允许发送原始消息：

- 当您为 Amazon Data Firehose 或亚马逊SQS终端节点启用原始消息传输时，将从已发布的消息中删除所有亚马逊SNS元数据，并按原样发送消息。
- 当您为 HTTP /S 端点启用原始消息传送时，其值设置为的HTTP标头`x-amz-sns-rawdelivery`将添加到消息中，表示消息已在未JSON格式化的情况下发布。`true`
- 当您为 HTTP /S 端点启用原始消息传输时，将传递消息正文、客户端 IP 和所需的标头。当您指定消息属性时，将不会发送它。
- 当您为 Firehose 端点启用原始消息传输时，消息正文将被传送。当您指定消息属性时，将不会发送它。

要使用启用原始消息传送 AWS SDK，必须使用`SetSubscriptionAttribute`API操作并将`RawMessageDelivery`属性的值设置为`true`。

利用 AWS Management Console实现原始消息传输

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 在主题页面上，选择已订阅到 Firehose、A SQS mazon HTTP 或 /S 终端节点的主题。
4. 在 ***MyTopic*** 页面上，在“订阅”部分中，选择订阅并选择编辑。
5. 在 E dit 中 ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***页面上的“详细信息”部分中，选择“启用原始消息传送”。
6. 选择 Save changes（保存更改）。

消息格式示例

在以下示例中，同一条消息被发送到同一 Amazon SQS 队列两次。唯一的区别是第一条消息禁用原始消息传输，第二条消息则启用该传输。

- 原始消息传输已禁用

```
{
  "Type": "Notification",
  "MessageId": "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
  "TopicArn": "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
  "Subject": "TestSubject",
  "Message": "This is a test message.",
  "Timestamp": "2021-02-16T21:41:19.978Z",
  "SignatureVersion": "1",
  "Signature":
 "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/
zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm
+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/
rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy
+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/
LsDr0Iak4ZDpg8dXg==",
  "SigningCertURL": "https://sns.us-east-2.amazonaws.com/
SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
```

```
  "UnsubscribeURL": "https://sns.us-east-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
}
```

- 原始消息传输已启用

```
This is a test message.
```

Amazon SQS 订阅的消息属性和原始消息传送

Amazon SNS 支持消息属性的传送，允许您提供有关消息的结构化元数据项，例如时间戳、地理空间数据、签名和标识符。对于启用了原始消息传送功能的 Amazon SQS 订阅，最多可以发送 10 个消息属性。要发送 10 个以上的消息属性，必须禁用 “原始消息传送”。但是，在启用原始消息传送的情况下，Amazon 会SNS丢弃定向到亚马逊SQS订阅的包含 10 个以上消息属性的消息，将其视为客户端错误。

使用其他账户向亚马逊SQS队列发送亚马逊SNS消息

本文档介绍如何通过在另一个账户中订阅一个或多个亚马逊SQS队列，向亚马逊SNS主题发布通知。如果主题和队列在同一账户下，那么您可以采用相同方法设置主题和队列（参阅 Fanout Amazon SNS 通知到亚马逊SQS队列进行异步处理）。主要区别在于您处理订阅确认的方式，这取决于您如何为队列订阅主题。

最佳做法是尽可能遵循队列拥有者创建订阅部分中引用的步骤，因为当队列拥有者创建订阅时会自动进行确认。

> **Note**
>
> 如果 Amazon SQS 队列的消息量很大，我们建议队列所有者创建订阅。

主题

队列拥有者创建订阅

创建 Amazon SQS 队列的账户是队列所有者。如果订阅由队列所有者创建，那么此订阅无需确认。一旦 Subscribe 操作完成后，队列即开始接收来自主题的通知。主题所有者必须提供队列所有者的账户权限，允许其对主题调用 Subscribe 操作，从而让队列所有者订阅主题所有者的主题。

步骤 1：使用 AWS Management Console设置主题策略

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 选择一个主题，然后选择 Edit（编辑）。
4. 在 E dit 中 ***MyTopic***页面上，展开 “访问策略” 部分。
5. 输入以下策略：

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "111122223333"
         },
         "Action": "sns:Subscribe",
         "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
   ]
}
```

此策略授予账户 111122223333 对账户 123456789012 中的 MyTopic 调用 sns:Subscribe 的权限。

具有账户 111122223333 的凭证的用户可以订阅 MyTopic。此权限允许账户 ID 向其IAM用户/角色委派权限。只有根账户或管理员用户才可以调用 sns:Subscribe。IAM用户/角色还必须sns:subscribe允许其队列进行订阅。

6. 选择 Save changes（保存更改）。

拥有账户凭证的用户111122223333可以订阅 MyTopic。

步骤 2： AWS 账户 使用 Amazon SQS 队列订阅添加到另一个主题中的某个主题 AWS Management Console

在开始之前，请确保您拥有主题和队列ARNs的权限，并且您已向该主题授予向队列发送消息的权限。

1. 登录 Amazon SQS 控制台。
2. 在导航窗格中，选择 Queues（队列）。
3. 从队列列表中选择要订阅 Amazon SNS 主题的队列。
4. 选择 “订阅 Amazon” SNS 主题。
5. 从可用于此队列的 “指定亚马逊” SNS 主题菜单中，为您的队列选择亚马逊SNS主题。
6. 选择 “输入亚马逊SNS主题”，ARN然后输入该主题的亚马逊资源名称 (ARN)。
7. 选择保存。

> **Note**
>
> - 为了能够与服务进行通信，队列必须具有 Amazon 的权限SNS。
> - 由于您是队列的所有者，因此您无需确认订阅。

非队列所有者用户创建订阅

创建订阅但不是队列所有者的任何用户都必须确认订阅。

当您使用该`Subscribe`操作时，Amazon SNS 会向队列发送订阅确认。订阅显示在 Amazon SNS 控制台中，其订阅 ID 设置为 “待确认”。

要确认订阅，有权从队列中读取消息的用户必须检索订阅确认URL，而订阅所有者必须使用订阅确认来确认订阅URL。确认订阅前，向主题发布的通知不会发送至队列。要确认订阅，您可以使用 Amazon SQS 控制台或`ReceiveMessage`操作。

> **Note**
>
> 在为终端节点订阅主题之前，请通过为队列设置 `sqs:SendMessage` 权限来确保队列可以接收来自主题的消息。有关更多信息，请参阅 第 2 步：向亚马逊SNS主题授予向亚马逊SQS队列发送消息的权限。

步骤 1： AWS 账户 使用 Amazon SQS 队列订阅添加到另一个主题中的某个主题 AWS Management Console

在开始之前，请确保您拥有主题和队列ARNs的权限，并且您已向该主题授予向队列发送消息的权限。

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择 Subscriptions（订阅）。
3. 在 Subscriptions（订阅）页面上，选择 Create subscription（创建订阅）。
4. 在 Create subscription（创建订阅）页上的 Details（详细信息）部分中，执行以下操作：
   a. 在 “主题” 中 ARNARN，输入主题的。
   b. 对于协议，请选择 Amazon SQS。
   c. 对于 EndpointARN，输入队列的。
   d. 选择创建订阅。

> **Note**
>
> - 为了能够与服务进行通信，队列必须具有 Amazon 的权限SNS。

以下是允许亚马逊SNS主题向亚马逊SQS队列发送消息的策略声明示例。

```
{
   "Sid": "Stmt1234",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "sqs:SendMessage",
   "Resource": "arn:aws:sqs:us-west-2:111111111111:QueueName",
   "Condition": {
      "ArnEquals": {
         "aws:SourceArn": "arn:aws:sns:us-west-2:555555555555:TopicName"
      }
   }
}
```

第 2 步：使用确认订阅 AWS Management Console

1. 登录 Amazon SQS 控制台。

2. 选择主题处于等待订阅阶段的队列。
3. 选择 Send and receive messages（发送和接收消息），然后选择 Poll for messages（轮询消息）。

   队列中会收到一条带有订阅确认的消息。
4. 在 Body（正文）列中，执行以下操作：

   a. 选择 More Details（更多详情）。

   b. 在 “消息详细信息” 对话框中，找到并记下 “订阅” URL 值。这是您的订阅链接（下面的示例）。有关API令牌验证的更多详情，请参阅ConfirmSubscription《Amazon SNS API参考》。

   ```
   https://sns.us-west-2.amazonaws.com/?
   Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
   east-2:123456789012:MyTopic&Token=2336412f37fb...
   ```

   c. 记下订阅确认链接。URL必须从队列所有者传递给订阅所有者。订阅所有者必须在 Amazon SNS 控制台中输入。URL
5. 以订阅所有者的身份登录亚马逊SNS控制台订阅所有者执行确认。
6. 选择相关主题。
7. 在主题的订阅清单表格中选择相关订阅。它被标记为“Pending confirmation”（等待确认）。
8. 选择 Confirm subscription（确认订阅）。
9. 将出现一个提示订阅确认链接的模态框。粘贴 订阅确认链接。
10. 在模态框中选择 Confirm subscription（确认订阅）。

    将显示一个XML响应，例如：

```
<ConfirmSubscriptionResponse>
   <ConfirmSubscriptionResult>
      <SubscriptionArn>arn:aws:sns:us-east-2:123456789012:MyTopic:1234a567-
bc89-012d-3e45-6fg7h890123i</SubscriptionArn>
   </ConfirmSubscriptionResult>
   <ResponseMetadata>
      <RequestId>abcd1efg-23hi-jkl4-m5no-p67q8rstuvw9</RequestId>
   </ResponseMetadata>
</ConfirmSubscriptionResponse>
```

    已订阅队列已准备好接收来自主题的消息。

11. （可选）如果您在 Amazon SNS 控制台中查看主题订阅，则可以看到 “订阅 ID” 列ARN中的待确认消息已被订阅所取代。

如何强制订阅要求对取消订阅请求进行身份验证？

订阅拥有者必须在订阅确认后将 `AuthenticateOnUnsubscribe` 标志设置为 true。

- 如果订阅由队列拥有者创建，则 `AuthenticateOnUnsubscribe` 自动设置为 true。
- 在没有身份验证的情况下导航到订阅确认链接时，无法将 `AuthenticateOnUnsubscribe` 设置为 true。

向不同区域的亚马逊SQS队列或 AWS Lambda 函数发送亚马逊SNS消息

Amazon SNS 支持跨区域配送，既适用于默认启用的区域，也适用于可选区域。有关亚马逊SNS支持的 AWS 区域的最新列表（包括可选区域），请参阅中的亚马逊简单通知服务终端节点和配额。Amazon Web Services 一般参考

Amazon SNS 支持跨区域向亚马逊SQS队列和 AWS Lambda 函数发送通知。当其中一个区域是可选区域时，您必须在订阅资源的政策中指定不同的亚马逊SNS服务委托人。

亚马逊SNS订阅命令必须在亚马逊SNS托管地区的目标账户中执行。例如，如果亚马逊在 us-SNS east-1 区域的账户 “A” 中，而 Lambda 函数位于 us-east-2 区域的账户 “B” 中，则订阅CLI命令必须在 us-east-1 区域的账户 “A” 中执行。

选择加入的区域

Amazon SNS 支持以下可选择加入区域：

区域名称	区域
非洲（开普敦）区域	af-south-1
亚太地区（香港）区域	ap-east-1
亚太地区（海得拉巴）区域	ap-south-2

区域名称	区域
亚太地区（雅加达）区域	ap-southeast-3
亚太地区（墨尔本）区域	ap-southeast-4
亚太地区（大阪）区域	ap-northeast-3
欧洲地区（米兰）	eu-south-1
欧洲地区（西班牙）区域	eu-south-2
欧洲（苏黎世）	eu-central-2
以色列（特拉维夫）区域	il-central-1
中东（巴林）区域	me-south-1
中东 (UAE) 区域	me-central-1

有关启用可选区域的信息，请参阅中的Amazon Web Services 一般参考管理 AWS 区域。

当您使用 Amazon SNS 将来自可选区域的消息传送到默认启用的区域时，您必须更改为队列创建的资源策略。将委托人 `sns.amazonaws.com` 替换为 `sns.<opt-in-region>.amazonaws.com`。例如：

- 要为美国东部（弗吉尼亚北部）的亚马逊SQS队列订阅亚太地区（香港）的亚马逊SNS主题，请将队列策略中的委托人更改为`sns.ap-east-1.amazonaws.com`。选择加入区域包括 2019 年 3 月 20 日之后推出的任何区域，包括亚太地区（香港）、亚太地区（雅加达）、中东（巴林）、欧盟（米兰）和非洲（开普敦）。2019 年 3 月 20 日之前推出的区域默认情况下处于启用状态。

  向 Amazon 提供跨区域配送支持 SQS

跨区域传输类型	支持/不支持	
原定设置启用的区域至选择加入区域	在队列的服务主体中使用 `sns.<opt-in-region>.amazonaws.com` 提供支持	
	在队列的服务主体中使用 `sns.<opt-in-region>.amazonaws.com` 提供支持	

跨区域传输类型	支持/不支持	
选择加入区域至原定设置启用的区域		
选择加入区域至选择加入区域	不支持	

以下是访问策略声明的示例，该声明允许选择加入区域 (af-south-1) 中的亚马逊SNS主题发送到某个区域 (us-east-1) 中的亚马逊SQS队列。enabled-by-default 它在路径 `Statement/Principal/Service` 下包含必要的区域化服务主体配置。

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "allow_sns_arn:aws:sns:af-south-1:111111111111:source_topic_name",
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.af-south-1.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "arn:aws:sqs:us-east-1:111111111111:destination_queue_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sns:af-south-1:111111111111:source_topic_name"
        }
      }
    },
    ...
  ]
}
```

- 要将美国东部（弗吉尼亚北部）的 AWS Lambda 函数订阅亚太地区（香港）的亚马逊SNS主题，请将 AWS Lambda 函数策略中的主体更改为`sns.ap-east-1.amazonaws.com`。选择加入区域包括 2019 年 3 月 20 日之后推出的任何区域，包括亚太地区（香港）、亚太地区（雅加达）、中东（巴林）、欧盟（米兰）和非洲（开普敦）。2019 年 3 月 20 日之前推出的区域默认情况下处于启用状态。

跨区域配送支持 AWS Lambda

跨区域传输类型	支持/不支持	
原定设置启用的区域至选择加入区域	不支持	
选择加入区域至原定设置启用的区域	在 Lambda 函数的服务主体中使用 `sns.<opt-in-region>.amazonaws.com` 提供支持	
选择加入区域至选择加入区域	不支持	

Amazon SNS 消息传送状态

Amaz SNS on 支持使用以下亚马逊SNS终端节点记录发送到主题的通知消息的发送状态：

- HTTP
- Amazon Data Firehose
- AWS Lambda
- 平台应用程序终端节点
- Amazon Simple Queue Service

配置消息传送状态属性后，发送给主题订阅者的消息的 CloudWatch 日志条目将发送到日志。记录消息传输状态有助于提供更好的业务洞察力，例如以下方面：

- 知道消息是否已传送到 Amazon SNS 终端节点。
- 识别从亚马逊SNS终端节点发送到亚马逊的响应SNS。
- 确定消息停留时间（从发布时间戳到传送到 Amazon SNS 终端节点之前的时间）。

要为消息传送状态配置主题属性，您可以使用 AWS 软件开发套件 (SDKs) API、查询或 AWS CloudFormation。 AWS Management Console

主题

使用 AWS Management Console配置传输状态日志记录

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 在主题页面上，选择主题，然后选择编辑。
4. 在 E dit 中 ***MyTopic***页面上，展开 “配送状态记录” 部分。
5. 选择要记录其传输状态日志的协议，例如 AWS Lambda。
6. 输入成功采样率（您希望接收 CloudWatch 日志的成功消息的百分比）。
7. 在 “IAM角色” 部分中，执行以下任一操作：

   - 要从您的账户中选择现有服务角色，请选择使用现有服务角色，然后为成功和失败的交付指定IAM角色。

   - 要在您的账户中创建新的服务角色，请选择创建新的服务角色，选择创建新角色以在IAM控制台中为成功和失败的交付定义IAM角色。

     要向 Amazon 授予代表您使用 CloudWatch 日志的SNS写入权限，请选择 “允许”。
8. 选择 Save changes（保存更改）。

   现在，您可以查看和解析包含消息传送状态的 CloudWatch 日志。有关使用的更多信息 CloudWatch，请参阅CloudWatch文档。

使用配置传送状态日志 AWS SDKs

AWS SDKs提供了多种语言版本，用于APIs在 Amazon 中使用消息传送状态属性SNS。

主题属性

您可以对消息传输状态使用下列主题属性名称值：

HTTP

- HTTPSuccessFeedbackRoleArn— 表示已订阅HTTP终端节点的 Amazon SNS 主题的成功消息传送状态。
- HTTPSuccessFeedbackSampleRate— 表示已订阅HTTP终端节点的 Amazon SNS 主题成功采样消息的百分比。
- HTTPFailureFeedbackRoleArn— 表示已订阅HTTP终端节点的 Amazon SNS 主题的消息传送失败状态。

Amazon Data Firehose

- FirehoseSuccessFeedbackRoleArn— 表示已订阅亚马逊 Kinesis Data Firehose 终端节点的亚马逊SNS主题的成功消息传输状态。
- FirehoseSuccessFeedbackSampleRate— 表示已订阅亚马逊 Kinesis Data Firehose 终端节点的亚马逊SNS主题成功采样消息的百分比。
- FirehoseFailureFeedbackRoleArn— 表示订阅了亚马逊 Kinesis Data Firehose 终端节点的亚马逊SNS主题的消息传输失败状态。

AWS Lambda

- LambdaSuccessFeedbackRoleArn— 表示已订阅 Lambda 终端节点的亚马逊SNS主题的成功消息传输状态。
- LambdaSuccessFeedbackSampleRate— 表示已订阅 Lambda 终端节点的 Amazon SNS 主题成功采样消息的百分比。
- LambdaFailureFeedbackRoleArn— 表示订阅 Lambda 终端SNS节点的亚马逊主题的消息传输失败状态。

平台应用程序终端节点

- ApplicationSuccessFeedbackRoleArn— 表示已订阅 AWS 应用程序终端节点的 Amazon SNS 主题的成功消息传送状态。
- ApplicationSuccessFeedbackSampleRate— 表示已订阅 AWS 应用程序终端节点的 Amazon SNS 主题成功采样消息的百分比。
- ApplicationFailureFeedbackRoleArn— 表示已订阅 AWS 应用程序终端节点的 Amazon SNS 主题的消息传送失败状态。

> **Note**
>
> 除了能够为发送到 Amazon SNS 应用程序终端节点的通知消息的消息传输状态配置主题属性外，您还可以为发送到推送通知服务的推送通知消息的传输状态配置应用程序属性。有关更多信息，请参阅使用 Amazon SNS 应用程序属性获取消息传送状态。

Amazon SQS

- SQSSuccessFeedbackRoleArn— 表示已订阅亚马逊SQS终端节点的亚马逊SNS主题的成功消息传送状态。
- SQSSuccessFeedbackSampleRate— 表示已订阅亚马逊SQS终端节点的亚马逊SNS主题成功采样消息的百分比。
- SQSFailureFeedbackRoleArn— 表示已订阅亚马逊SQS终端节点的亚马逊SNS主题的消息传送失败状态。

> **Note**
>
> <ENDPOINT>SuccessFeedbackRoleArn和<ENDPOINT>FailureFeedbackRoleArn属性用于向 Amazon 授予代表您使用 CloudWatch 日志的SNS写入权限。<ENDPOINT>SuccessFeedbackSampleRate 属性用于指定成功传输消息的采样率百分比 (0-100)。配置该<ENDPOINT>FailureFeedbackRoleArn属性后，所有失败的消息传送都会生成 CloudWatch 日志。

AWS SDK配置主题属性的示例

以下代码示例演示如何使用 SetTopicAttributes。

CLI

AWS CLI

为主题设置属性

以下 set-topic-attributes 示例为指定主题设置 DisplayName 属性。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

此命令不生成任何输出。

- 有关API详细信息，请参阅 “SetTopicAttributes AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>
```

```
                Where:
                   attribute - The attribute action to use. Valid parameters are:
Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

       if (args.length < 3) {
           System.out.println(usage);
           System.exit(1);
       }

       String attribute = args[0];
       String topicArn = args[1];
       String value = args[2];

       SnsClient snsClient = SnsClient.builder()
               .region(Region.US_EAST_1)
               .build();

       setTopAttr(snsClient, attribute, topicArn, value);
       snsClient.close();
   }

   public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {
       try {
           SetTopicAttributesRequest request =
SetTopicAttributesRequest.builder()
                   .attributeName(attribute)
                   .attributeValue(value)
                   .topicArn(topicArn)
                   .build();

           SetTopicAttributesResponse result =
snsClient.setTopicAttributes(request);
           System.out.println(
                   "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
"\n\nTopic " + request.topicArn()
                           + " updated " + request.attributeName() + " to " +
request.attributeValue());

       } catch (SnsException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 SetTopicAttributes” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
```

```
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 SetTopicAttributes” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
```

```
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- 有关API详细信息，请参阅SetTopicAttributes中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
```

```
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 SetTopicAttributes” 中的。

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
```

```
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: "2008-10-17",
      Id: "__default_policy_ID",
      Statement: [{
                    Sid: "__default_statement_ID",
                    Effect: "Allow",
                    Principal: { "AWS": "*" },
                    Action: ["SNS:Publish"],
                    Resource: topic_arn,
                    Condition: {
                      ArnEquals: {
                        "AWS:SourceArn": resource_arn
                      }
                    }
                  }]
    }.to_json
  end
end

# Example usage:
```

```
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN"     # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME"    # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for Ruby API参考 SetTopicAttributes” 中的。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        lo_sns->settopicattributes(
            iv_topicarn = iv_topic_arn
            iv_attributename  = iv_attribute_name
            iv_attributevalue = iv_attribute_value
        ).
        MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅SetTopicAttributes中的AWS SDK以供SAPABAPAPI参考。

使用 AWS CloudFormation配置传输状态日志记录

要DeliveryStatusLogging使用进行配置 AWS CloudFormation，请使用JSON或YAML模板创建 AWS CloudFormation 堆栈。有关更多信息，请参阅《 AWS CloudFormation 用户指南》中的AWS::SNS::Topic资源DeliveryStatusLogging属性。以下是JSON和中的 AWS CloudFormation 模板示例YAML，用于创建新主题或使用亚马逊SQS协议的所有DeliveryStatusLogging属性更新现有主题。

JSON

```
"Resources": {
    "MySNSTopic" : {
        "Type" : "AWS::SNS::Topic",
        "Properties" : {
            "TopicName" : "TestTopic",
            "DisplayName" : "TEST",
            "SignatureVersion" : "2",
            "DeliveryStatusLogging" : [{
                "Protocol": "sqs",
                "SuccessFeedbackSampleRate": "45",
                "SuccessFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1",
                "FailureFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2"
            }]
        }
    }
}
```

YAML

```
Resources:
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      TopicName:TestTopic
      DisplayName:TEST
      SignatureVersion:2
      DeliveryStatusLogging:
       - Protocol: sqs
         SuccessFeedbackSampleRate: 45
```

```
        SuccessFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1
        FailureFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2
```

Amazon SNS 消息传送重试次数

Amazon 为每种交付协议SNS定义了配送政策。传输策略定义了当服务器端出现错误时（托管订阅终端节点的系统不可用时），Amazon 如何SNS重试消息传送。当配送策略用尽后，Amazon 将SNS停止重试传送并丢弃消息，除非订阅中附加了死信队列。有关更多信息，请参阅 Amazon SNS 死信队列。

主题

传输协议和策略

> **Note**
>
> - 除了 HTTP /S 之外，您无法更改亚马逊SNS定义的配送政策。只有 HTTP /S 支持自定义策略。请参阅 创建 HTTP /S 传输策略。
> - Amazon 对配送重试SNS采用抖动。有关更多信息，请参阅发布在 AWS 架构博客上的指数回退和抖动博客文章。
> - HTTP/S 终端节点的总策略重试时间不能超过 3,600 秒。这是一项硬性限制，无法增加。

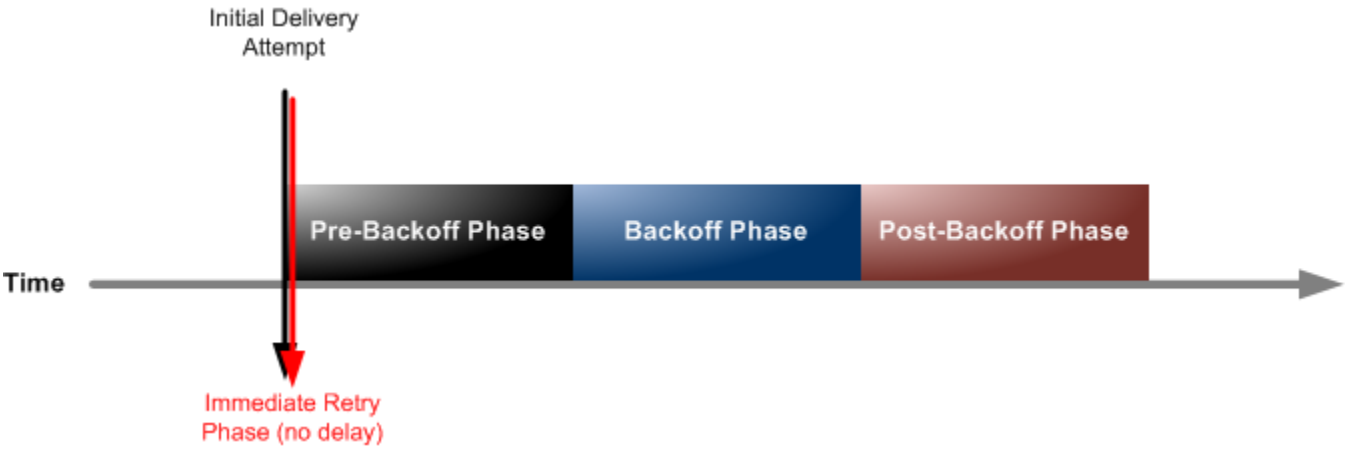
端点类型	传输协议	立即重试（无延迟）阶段	前退避阶段	退避阶段	后退避阶段	总尝试次数
AWS 托管端点	Amazon Data Firehose[1]	3 次，无延迟	2 次，相隔 1 秒	10 次，带指数退避	100000 次，相隔 20 秒	100015 次，超过 23 天

端点类型	传输协议	立即重试（无延迟）阶段	前退避阶段	退避阶段	后退避阶段	总尝试次数
	AWS Lambda			（1 秒到 20 秒）		
	Amazon SQS					
客户托管的终端节点	SMTP	0 次，无延迟	2 次，相隔 10 秒	10 次，带指数退避（10 秒到 600 秒（10 分钟））	38 次，相隔 600 秒（10 分钟）	50 次尝试，超过 6 小时
	SMS					
	移动推送					

[1] 对于 Firehose 协议的限制错误，亚马逊SNS使用与客户托管终端节点相同的交付政策。

传输策略阶段

下图显示了传输策略的各个阶段。



每项交付政策由四个阶段组成：

1. 立即重试阶段（无延迟）– 此阶段在首次传输尝试结束后立即发生。在该阶段中重试之间没有延迟。
2. 前退避阶段 – 此紧随即刻重试阶段。在应用退避功能之前，Amazon SNS 使用此阶段尝试一系列重试。此阶段指定重试次数以及它们之间的延迟时间量。

3. 退避阶段 – 此阶段通过使用重试-退避函数控制各个重试之间的延迟。此阶段设置了最短延迟时间、最长延迟时间和重试-退避函数，该函数定义了延迟时间从最小值增加到最大值的速度。退避函数可以是算术、指数、几何或线性的。
4. 后退避阶段 – 此阶段在退避阶段之后发生。此阶段指定重试次数以及它们之间的延迟时间量。它是最后一个阶段。

创建 HTTP /S 传输策略

您可以使用传输策略及其四个阶段来定义 Amazon 如何SNS重试将消息传送到 HTTP /S 终端节点。例如，当您可能想要根据HTTP服务器容量自定义HTTP终端节点策略时，Amazon SNS 允许您覆盖终端节点的默认重试策略。

您可以将 HTTP /S 传输策略设置为订阅或主题级别的JSON对象。在主题级别定义策略时，它适用于与该主题关联的所有 HTTP /S 订阅。要在订阅级别设置传送政策，您可以使用`Subscribe`或`SetSubscriptionAttributes`API操作。要在主题级别设置传输策略，您可以使用`CreateTopic`或`SetTopicAttributes`API操作。或者，您也可以在 AWS CloudFormation 模板中使用AWSSNS::: 订阅资源。

您应该根据您的 HTTP /S 服务器的容量自定义您的传输策略。可以将策略设置为主题属性或订阅属性。如果您的主题中的所有 HTTP /S 订阅都针对同一 HTTP /S 服务器，我们建议您将传送策略设置为主题属性，使其对主题中的所有 HTTP /S 订阅仍然有效。否则，您必须根据策略所针对的 HTTP /S 服务器的容量，为主题中的每个 HTTP /S 订阅制定传输策略。

您还可以在请求策略中设置 Content-Type 标头，以指定通知的媒体类型。默认情况下，Amazon SNS 会将所有通知发送到内容类型设置为 HTTP /S 的终端节点`text/plain; charset=UTF-8`。Amazon SNS 允许您改写默认请求策略。有关支持的 `headerContentType` 和约束，请参阅下表。

以下JSON对象表示一项配送策略，该策略指示 Amazon SNS 重试失败的 HTTP /S 配送尝试，如下所示：

1. 在无延迟阶段立即尝试 3 次
2. 在前退避阶段尝试 2 次（相隔 1 秒）
3. 10 次（带指数退避，1 秒到 60 秒）
4. 在后退避阶段尝试 35 次（相隔 60 秒）

在此示例传送政策中，Ama SNS zon 在丢弃消息之前总共尝试了 50 次。要在传送策略中指定的重试次数用尽后保留消息，请将您的订阅配置为将无法投递的消息移至死信队列 ()。DLQ有关更多信息，请参阅 Amazon SNS 死信队列。

> **Note**
>
> 该配送政策还指示亚马逊SNS使用该物业将配送限制在每秒不超过10次。maxReceivesPerSecond这种自限制速率可能导致发布的消息（入站流量）多于已发送的消息（出站流量）。当入站流量多于出站流量时，您的订阅可能会累积大量的消息积压，从而可能会导致较长的消息传输延迟。在传输策略中，请确保为 maxReceivesPerSecond 指定一个不会对您的工作负载产生不利影响的值。

> **Note**
>
> 此传送策略覆盖了 HTTP /S 通知的默认内容类型。application/json

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

传送策略包含重试策略、节流策略和请求策略。传送策略共有 9 个属性。

策略	描述	限制
minDelayTarget	重试的最短延迟时间。 单位：秒	1 至最长延迟时间 原定设置值：20
maxDelayTarget	重试的最长延迟时间。 单位：秒	最短延迟时间至 3600 原定设置值：20
numRetries	重试总数，包括立即重试、前退避重试、退避重试和后退避重试。	0 至 100 原定设置值：3
numNoDelayRetries	要立即完成的重试次数，各个重试之间无延迟。	0 或更多 原定设置值：0
numMinDelayRetries	前退避阶段的重试次数，各个重试之间有指定的最短延迟时间。	0 或更多 原定设置值：0
numMaxDelayRetries	后退避阶段的重试次数，各个重试之间有最长延迟时间。	0 或更多 原定设置值：0
backoffFunction	各个重试之间退避的模型。	四个选项之一： • 算术 • 指数 • 几何 • 线性 默认：线性
maxReceivesPerSecond	每个订阅每秒的最大传输次数。	1 或更多 默认：无限制

策略	描述	限制
headerContentType	发送到 HTTP /S 端点的通知的内容类型。	如果未定义请求策略，则内容类型原定设置为 text/plain; charset=UTF-8 。 当为订阅禁用原始消息传送时（原定设置），或者在主题级别定义传送策略时，支持的标头内容类型为 application/json 和 text/plain 。 为订阅启用原始消息传送时，支持以下内容类型： • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • application/octet-stream • application/soap+xml • 应用程序/ x-www-form-urlencoded • application/xhtml+xml • application/xml

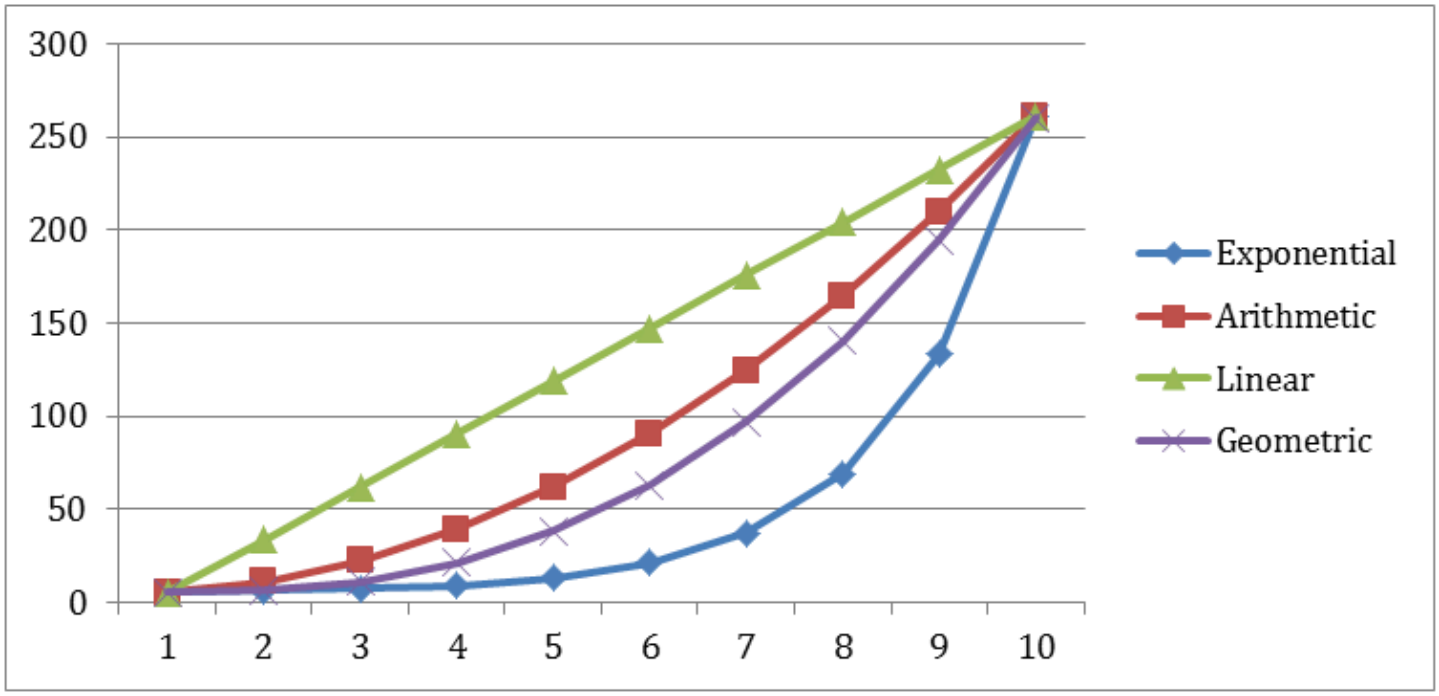
Amazon SNS 使用以下公式计算退避阶段的重试次数：

```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

您可以使用三个参数来控制退避阶段的重试频率。

- `minDelayTarget` – 定义了在退避阶段中与第一次重试尝试关联的延迟时间。
- `maxDelayTarget` – 定义了在退避阶段中与最后一次重试尝试关联的延迟时间。
- `backoffFunction`— 定义 Amazon SNS 用于计算退避阶段第一次和最后一次重试之间的所有重试尝试相关的延迟的算法。您可以从四个重试-退避函数中进行选择。

下图显示了每个重试退避函数如何影响退避阶段中与重试关联的延迟时间：一种传输策略，其重试总数设置为 10，最短延迟时间设置为 5 秒，最长延迟时间设置为 260 秒。纵轴以秒为单位表示与 10 次重试中的每个重试相关的延迟。水平轴表示从第 1 次尝试到第 10 次尝试的重试次数。



Amazon SNS 死信队列

死信队列是亚马逊SNS订阅可以针对无法成功发送给订阅者的消息的亚马逊SQS队列。由于客户端错误或服务器错误而无法传输的消息将保留在死信队列中，以进行进一步分析或重新处理。有关更多信息，请参阅为订阅配置 Amazon SNS 死信队列 和Amazon SNS 消息传送重试次数。

> **Note**
>
> - 亚马逊SNS订阅和亚马逊SQS队列必须位于相同的 AWS 账户和区域下。
> - 对于FIFO主题，您可以使用亚马逊SQS队列作为亚马逊订阅的死信队列。SNSFIFO主题订阅使用FIFO队列，标准主题订阅使用标准队列。

- 要将加密的 Amazon SQS 队列用作死信队列，您必须使用KMS带有密钥策略的自定义队列，该策略允许亚马逊SNS服务委托人访问 AWS KMS API操作。有关更多信息，请参阅使用服务器端加密保护 Amazon SNS 数据本指南和使用服务器端加密保护亚马逊SQS数据 (SSE) 和 AWS KMS《亚马逊简单队列服务开发者指南》。

主题

为什么消息传输会失败？

通常，当 Amazon 由于客户端或服务器端错误而SNS无法访问已订阅的终端节点时，消息传递就会失败。当 Amazon SNS 收到客户端错误或继续收到超过相应重试策略规定的重试次数的消息的服务器端错误时，Amazon 会SNS丢弃该消息，除非订阅中附加了死信队列。失败的传输不会更改您的订阅状态。有关更多信息，请参阅 Amazon SNS 消息传送重试次数。

客户端错误

当 Amazon 的订阅元数据SNS过时时，可能会发生客户端错误。这些错误通常发生在所有者删除终端节点（例如，订阅了 Amazon SNS 主题的 Lambda 函数），或者所有者更改已订阅终端节点的策略以SNS阻止亚马逊向该终端节点发送消息时。Amazon SNS 不会重试因客户端错误而失败的消息传送。

服务器端错误

当负责订阅终端节点的系统不可用或返回异常表示无法处理来自 Amazon 的有效请求时，可能会发生服务器端错误。SNS当服务器端出现错误时，Amazon 会使用线性或指数退避函数SNS重试失败的交付。对于由亚马逊支持的 AWS 托管终端节点导致的服务器端错误 AWS Lambda，亚马逊SQS将在 23 天内SNS重试投递多达 100,015 次。

客户管理的端点（例如HTTP、SMTPSMS、或移动推送）也可能导致服务器端错误。Amazon 也会SNS重试向这些类型的终端节点传送。虽然HTTP终端节点支持客户定义的重试策略，但对于和移动推送终端节点，Amazon 将内部传输重试策略SNS设置为SMTP在 6 小时内 50 次。SMS

死信队列的工作方式

Amazon SNS 订阅（而不是主题）会附加死信队列，因为消息是在订阅级别进行的。这使您能够更轻松地识别每条消息的原始目标终端节点。

与亚马逊SNS订阅相关的死信队列是普通的亚马逊SQS队列。有关消息保留期的更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中与消息相关的配额。您可以使用 Amazon SQS `SetQueueAttributes` API 操作更改消息保留期。为了使您的应用程序更具弹性，我们建议将死信队列的最长保留期设置为 14 天。

如何将消息移至死信队列中？

可以使用重新驱动策略 将您的消息移至死信队列中。重新驱动策略是一个引用死信队列ARN的JSON对象。该`deadLetterTargetArn`属性指定ARN。ARN必须指向与您的亚马逊订SNS阅位于相同 AWS 账户 和区域的亚马逊SQS队列。有关更多信息，请参阅 为订阅配置 Amazon SNS 死信队列。

以下JSON对象是附加到SNS订阅的续订政策示例。

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

如何将消息移出死信队列？

可以通过两种方式将消息移出死信队列：

- 避免编写 Amazon 使用SQS者逻辑 — 将死信队列设置为 Lambda 函数的事件源，以耗尽死信队列。
- 编写 Amazon 使用SQS者逻辑 — 使用 Amazon SQS API AWS SDK、或 AWS CLI 编写自定义使用者逻辑，用于轮询、处理和删除死信队列中的消息。

如何监控和记录死信队列？

您可以使用亚马逊 CloudWatch 指标来监控与您的亚马逊订阅相关的死信队列。SNS所有 Amazon SQS 队列每隔一分钟发布一次 CloudWatch 指标。有关更多信息，请参阅《亚马逊简单队列服务开发者指南》SQS中的亚马逊可用 CloudWatch 指标。所有带有死信队列的 Amazon SNS 订阅也会发布指标。 CloudWatch 有关更多信息，请参阅 使用监控亚马逊SNS话题 CloudWatch。

要收到死信队列中的活动通知，您可以使用 CloudWatch 指标和警报。为该NumberOfMessagesSent指标设置警报是不合适的，因为该指标不会捕获因处理尝试失败而发送给的消息。DLQ取而代之的是使用ApproximateNumberOfMessagesVisible指标，它可以捕获中当前可用的所有消息DLQ，包括因处理失败而移动的消息。

CloudWatch 警报设置示例

1. 为该**ApproximateNumberOfMessagesVisible**指标创建CloudWatch警报。
2. 将警报阈值设置为 1（或根据您的期望和DLQ流量设置其他适当的值）。
3. 指定警报响起时要通知的 Amazon SNS 主题。此 Amazon SNS 主题可以将您的警报通知发送到任何终端节点类型（例如电子邮件地址、电话号码或移动寻呼机应用程序）。

您可以使用 CloudWatch 日志来调查导致任何 Amazon SNS 交付失败的异常，以及将消息发送到死信队列。Amazon SNS 可以记录成功和失败的交付 CloudWatch。有关更多信息，请参阅 Amazon SNS 移动应用程序属性。

为订阅配置 Amazon SNS 死信队列

死信队列是亚马逊SNS订阅可以针对无法成功发送给订阅者的消息的亚马逊SQS队列。由于客户端错误或服务器错误而无法传输的消息将保留在死信队列中，以进行进一步分析或重新处理。有关更多信息，请参阅Amazon SNS 死信队列 和Amazon SNS 消息传送重试次数。

本页展示了如何使用 AWS Management Console、、 AWS CLI、和 AWS CloudFormation 来为 Ama SNS zon 订阅配置死信队列。 AWS SDK

> **Note**
>
> 对于FIFO主题，您可以使用亚马逊SQS队列作为亚马逊订阅的死信队列。SNSFIFO主题订阅使用FIFO队列，标准主题订阅使用标准队列。

先决条件

在配置死信队列之前，请完成以下先决条件：

1. 创建名为的 Amazon SNS 主题MyTopic。
2. 创建名为的亚马逊SQS队列MyEndpoint，用作亚马逊SNS订阅的终端节点。
3. （跳过 AWS CloudFormation）在队列中订阅主题。

4. 创建另一个名为 Amazon SQS 队列MyDeadLetterQueue，用作亚马逊SNS订阅的死信队列。
5. 要授予亚马逊SNS委托人访问亚马逊SQSAPI操作的权限，请为设置以下队列策略MyDeadLetterQueue。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
    }
  }]
}
```

主题

- 要使用 Amazon SNS 订阅配置死信队列 AWS Management Console
- 使用为 Amazon SNS 订阅配置死信队列 AWS SDK
- 要使用 Amazon SNS 订阅配置死信队列 AWS CLI
- 使用为 Ama SNS zon 订阅配置死信队列 AWS CloudFormation

要使用 Amazon SNS 订阅配置死信队列 AWS Management Console

在开始本教程之前，请确保完成先决条件。

1. 登录 Amazon SQS 控制台。
2. 创建 Amazon SQS 队列或使用现有队列并在队列的 “详情” 选项卡上记下该队列的内容，例如：ARN

```
arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue
```

3. 登录 Amazon SNS 控制台。
4. 在导航面板中，选择 Subscriptions（订阅）。

5. 在 Subscriptions (订阅) 页面上，选择现有订阅，然后选择 Edit (编辑)。
6. 在 E dit 中 ***1234a567-bc89-012d-3e45-6fg7h890123i***页面上，展开 Redrive 策略（死信队列）部分，然后执行以下操作：
   a. 选择 Enabled (已启用)。
   b. 指定 A ARN mazon SQS 队列的。
7. 选择 Save changes（保存更改）。

   您的订阅将配置为使用死信队列。

使用为 Amazon SNS 订阅配置死信队列 AWS SDK

在您运行此示例之前，请确保完成先决条件。

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置AWS SDKs和凭据文件。

以下代码示例显示了如何使用`SetSubscriptionAttributesRedrivePolicy`。

Java

SDK适用于 Java 1.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
```

```
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

要使用 Amazon SNS 订阅配置死信队列 AWS CLI

在开始本教程之前，请确保完成先决条件。

1. 安装和配置 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南。
2. 使用以下命令。

```
aws sns set-subscription-attributes \
--subscription-arn arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i
--attribute-name RedrivePolicy
--attribute-value "{\"deadLetterTargetArn\": \"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}"
```

使用为 Ama SNS zon 订阅配置死信队列 AWS CloudFormation

在开始本教程之前，请确保完成先决条件。

1. 将以下JSON代码复制到名为的文件中MyDeadLetterQueue.json。

```
{
  "Resources": {
    "mySubscription": {
      "Type" : "AWS::SNS::Subscription",
      "Properties" : {
        "Protocol": "sqs",
        "Endpoint": "arn:aws:sqs:us-east-2:123456789012:MyEndpoint",
        "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
        "RedrivePolicy": {
          "deadLetterTargetArn":
            "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
```

```
        }
      }
    }
  }
}
```

2. 登录 AWS CloudFormation 控制台。
3. 在 Select Template (选择模板) 页面上，选择 Upload a template to Amazon S3 (将模板上传到 Amazon S3)，再选择您的 `MyDeadLetterQueue.json` 文件，然后选择 Next (下一步)。
4. 在 Specify Details（指定详细信息）页面上，为 Stack Name（堆栈名称）输入 `MyDeadLetterQueue`，然后选择 Next（下一步）。
5. 在选项页面上，选择下一步。
6. 在 Review 页面上，选择 Create 。

   AWS CloudFormation 开始创建`MyDeadLetterQueue`堆栈并显示 CREATE_IN_ 状态 PROGRESS。该过程完成后， AWS CloudFormation 将显示 CREATE_ COMPLETE 状态。

Amazon SNS 消息存档、重播和分析

亚马逊SNS标准主题支持通过 Amazon Data Firehose 存档消息。你可以将通知分散到Firehose传送流，这样你就可以将通知发送到Firehose支持的存储和分析目的地，包括亚马逊简单存储服务(Amazon S3)、Amazon Redshift等。

Amazon SNS FIFO 主题支持就地、无代码的消息存档，允许主题所有者存储（或存档）向主题发布的消息长达 365 天。对于具有活动 `ArchivePolicy` 的主题，订阅用户可以创建 `ReplayPolicy`，以将归档的消息检索（或重播）回订阅的端点。要了解有关此特征的更多信息，请参阅Amazon SNS 消息存档和重播FIFO主题。

特征	标准主题	FIFO话题
消息归档	Fanout Amazon SNS 通知，带有 Firehose 传送流，可增强数据管理	为FIFO主题所有者提供的 Amazon SNS 消息存档
消息重播	标准主题的重播不是内置特征。许多客户根据自己的消息归档来构建自己的重播。	为FIFO主题订阅者提供的 Amazon SNS 消息重播

Amazon 中的资源管理和优化 SNS

使用亚马逊SNS控制台创建和配置亚马逊SNS主题和订阅。有关 Amazon 的更多信息SNS，请参阅 什么是亚马逊SNS？

主题

- Amazon SNS 话题标记

Amazon SNS 话题标记

亚马逊SNS支持为亚马逊SNS话题添加标签。这可以帮助您跟踪和管理与主题相关的成本，增强您的 Ident AWS ity and Access Management (IAM) 策略的安全性，并允许您轻松搜索或筛选数千个主题。通过添加标签，您可以使用 AWS 资源组（Resource Groups）管理您的亚马逊SNS主题。有关资源组的更多信息，请参阅AWS 资源组用户指南。

主题

- 成本分配的标记
- 访问控制的标记
- 进行标记以便进行资源搜索和筛选
- 配置 Amazon SNS 主题标签

成本分配的标记

要整理和确定您的 Amazon SNS 主题以进行成本分配，您可以添加标识主题目的的标签。这在您拥有许多主题时尤其有用。您可以使用成本分配标签来整理 AWS 账单，以反映您自己的成本结构。为此，请注册以获取包含标签密钥和值的 AWS 账户账单。有关更多信息，请参阅 AWS 账单和成本管理用户指南中的设置月度成本分配报告。

例如，您可以添加代表成本中心和亚马逊SNS主题目的的标签，如下所示：

资源	键	值
主题 1	成本中心	43289
	应用程序	订单处理

资源	键	值
主题 2	成本中心	43289
	应用程序	支付处理
主题 3	成本中心	76585
	应用程序	存档

此标记方案可让您将执行相关任务的两个主题分组到同一成本中心，并使用不同的成本分配标签来标记不相关的活动。

访问控制的标记

AWS Identity and Access Management 支持根据标签控制对资源的访问权限。标记资源后，在IAM策略的条件元素中提供有关资源标签的信息，以管理基于标签的访问权限。有关如何使用 Amazon SNS 控制台或为资源添加标签的信息 AWS SDK，请参阅配置标签。

您可以限制IAM身份的访问权限。例如，您可以限制Publish和PublishBatch访问包含带有密钥environment和值的标签的所有亚马逊SNS主题production，同时允许访问所有其他亚马逊SNS主题。在下面的示例中，该策略限制了将消息发布到标记为的主题的能力production，同时允许将消息发布到标记为的主题development。有关更多信息，请参阅《IAM用户指南》中的使用标签控制访问权限。

> **Note**
>
> 为设置IAM权限将同时Publish设置Publish和的权限PublishBatch。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": [
 "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
```

```
      "StringEquals": {
        "aws:ResourceTag/environment": "production"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "development"
      }
    }
  }]
}
```

进行标记以便进行资源搜索和筛选

一个 AWS 账户可以有成千上万的亚马逊SNS话题（详情请参阅亚马逊SNS配额）。通过标记主题，您可以简化搜索或筛选主题的过程。

例如，您可以拥有数百个与您的生产环境相关的主题。您可以通过给定标签查询所有主题，而不必手动搜索这些主题：

```
import com.amazonaws.services.resourcegroups.AWSResourceGroups;
import com.amazonaws.services.resourcegroups.AWSResourceGroupsClientBuilder;
import com.amazonaws.services.resourcegroups.model.QueryType;
import com.amazonaws.services.resourcegroups.model.ResourceQuery;
import com.amazonaws.services.resourcegroups.model.SearchResourcesRequest;
import com.amazonaws.services.resourcegroups.model.SearchResourcesResult;

public class Example {
    public static void main(String[] args) {
        // Query Amazon SNS Topics with tag "keyA" as "valueA"
        final String QUERY = "{\"ResourceTypeFilters\":[\"AWS::SNS::Topic\"],
\"TagFilters\":[{\"Key\":\"keyA\", \"Values\":[\"valueA\"]}]}";

        // Initialize ResourceGroup client
        AWSResourceGroups awsResourceGroups = AWSResourceGroupsClientBuilder
            .standard()
```

```
            .build();

        // Query all resources with certain tags from ResourceGroups
        SearchResourcesResult result = awsResourceGroups.searchResources(
            new SearchResourcesRequest().withResourceQuery(
                new ResourceQuery()
                .withType(QueryType.TAG_FILTERS_1_0)
                .withQuery(QUERY)
            ));
        System.out.println("SNS Topics with certain tags are " +
 result.getResourceIdentifiers());
    }
}
```

配置 Amazon SNS 主题标签

本页展示了如何使用 AWS Management Console AWS SDK、和 ， AWS CLI为亚马逊SNS主题配置标签。

> **Important**
>
> 请勿在标签中添加个人身份信息 (PII) 或其他机密或敏感信息。标签可供许多其他亚马逊云科技访问，包括计费。标签不适合用于私有或敏感数据。

主题

- 使用 Amazon SNS 主题列出、添加和移除标签 AWS Management Console
- 使用向主题添加标签 AWS SDK
- 使用 Amazon SNS API 操作管理标签
- API支持的动作 ABAC

使用 Amazon SNS 主题列出、添加和移除标签 AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 在主题页面上，选择主题，然后选择编辑。
4. 展开标签部分。

将列出添加到主题的标签。

5. 修改主题标签：
   - 要添加标签，请选择 Add tag（添加标签），然后输入 Key（键）和 Value（值）（可选）。
   - 要删除标签，请选择键值对旁边的删除标签。
6. 选择 Save changes（保存更改）。

使用向主题添加标签 AWS SDK

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置AWS SDKs和凭据文件。

以下代码示例演示如何使用 `TagResource`。

CLI

AWS CLI

为主题添加标签

以下`tag-resource`示例向指定的 Amazon SNS 主题添加元数据标签。

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

此命令不生成任何输出。

- 有关API详细信息，请参阅 “TagResource AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
```

```
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 TagResource” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

    val tag2 =
        Tag {
            key = "Environment"
            value = "Gamma"
        }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request =
        TagResourceRequest {
            resourceArn = topicArn
            tags = tagList
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- 有关API详细信息，请参阅TagResource中的 Kotlin AWS SDK API 参考。

使用 Amazon SNS API 操作管理标签

要使用 Amazon 管理标签 SNSAPI，请使用以下API操作：

- ListTagsForResource
- TagResource
- UntagResource

API支持的动作 ABAC

以下是支持基于属性的访问控制 () ABAC 的API操作列表。有关的更多详细信息ABAC，请参阅ABAC 用途是什么 AWS？ 在《IAM用户指南》中。

- AddPermission
- ConfirmSubscription
- DeleteTopic
- GetDataProtectionPolicy
- GetSubscriptionAttributes
- GetTopicAttributes
- ListSubscriptionsByTopic
- ListTagsForResource
- Publish
- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource

Amazon SNS 事件来源和目的地

Amazon 通过路由事件驱动的通知来SNS连接 AWS 服务 外部系统。Amazon SNS 接收来自各种事件 AWS 服务，例如数据管道更新、Amazon EC2 扩展操作或安全警报，并将这些事件发布到亚马逊SNS主题。然后，这些主题会将通知发送到指定的目的地。

Amazon SNS 支持两种主要类型的目标：应用程序到应用程序 (A2A) 和应用程序到个人 (A2P)。在 A2A 消息中，亚马逊SNS可以向 Lambda 发送事件以触发自定义业务逻辑，向亚马逊发送事件以排队消息，向亚马SQS逊 Kinesis Data Firehose 发送事件以将数据流式传输到存储和分析服务。对于 A2P 消息，Amazon SNS 可以通过SMS电子邮件向移动设备发送通知和推送通知，从而确保用户或团队及时收到警报。

通过充当中心枢纽，Amazon SNS 将通知发送到正确的位置，帮助您更有效地自动化和管理 AWS 基础设施。这种设置允许服务之间的无缝集成以及与用户和系统的可靠通信。

主题

Amazon SNS 事件来源

本主题列出了可以 AWS 向 Amazon SNS 主题发布事件的服务，按其AWS 产品类别分组。

> **Note**
>
> 亚马逊在 2020 年 10 月SNS推出了FIFO话题。目前，大多数 AWS 服务仅支持向标准主题发送事件。

主题

- 容器服务
- 客户参与服务
- 数据库服务
- 开发人员工具服务
- 前端 Web 和移动服务
- 游戏开发服务
- 物联网服务
- 机器学习服务
- 管理和治理服务
- 媒体服务
- 迁移和传输服务
- 网络和内容分发服务
- 安全性、身份与合规性服务
- 无服务器服务
- 存储服务
- 其他事件来源

分析服务

下表描述了亚马逊如何SNS与 Athena 和 Amazon Redshift 等 AWS 分析服务集成 AWS Data Pipeline，为关键事件提供实时通知，包括违反控制限制、管道状态更新和数据仓库活动。

您可以利用这些集成来自动响应并保持对数据操作的有效监督。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon Athena — 允许您使用标准分析亚马逊 S3 中的数据。SQL	在超出控制限制时接收通知。有关更多信息，请参阅 Amazon Athena 用户指南中的设置数据使用控制限制。
AWS Data Pipeline – 帮助自动执行数据的移动与转换。	接收有关管道组件状态的通知。有关更多信息，请参阅SnsAlarm《AWS Data Pipeline 开发人员指南》。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon Redshift – 管理数据仓库的所有设置、操作和扩展工作。	接收 Amazon Redshift 事件的通知。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 事件通知。

应用程序集成服务

下表介绍了 Amazon 如何与应用程序SNS集成服务（例如 EventBridge 和）集成 AWS Step Functions，从而为关键业务应用程序启用实时数据路由和通知。

您可以利用这些集成来接收来自 EventBridge 事件的警报，并使用 Step Functions 编排工作流程，从而增强应用程序的自动化和响应能力。

AWS 服务	在 Amazon 上使用的好处 SNS
亚马逊 EventBridge — 提供来自您自己的应用程序、 software-as-a-service (SaaS) 应用程序的实时数据流， AWS 服务 并将这些数据传送到目标，包括亚马逊SNS。 EventBridge 以前叫做“ CloudWatch 活动”。	接收 EventBridge 事件通知。有关更多信息，请参阅《亚马逊 EventBridge 用户指南》中的亚马逊 EventBridge目标。
AWS Step Functions— 允许您组合 AWS Lambda 功能和其他功能 AWS 服务 来构建关键业务应用程序。	接收 Step Functions 事件的通知。有关更多信息，请参阅AWS Step Functions 开发者指南中的SNS使用 Step Functions 致电亚马逊。

账单和成本管理服务

下表描述了如何 AWS Billing and Cost Management 与 Amazon 集成，SNS以提供预算、价格变动和成本异常通知。

您可以利用此集成来设置Amazon SNS 主题，以接收有关您的 AWS 支出的实时提醒，从而帮助您监控成本并有效地应对意外费用。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS Billing and Cost Management – 提供帮助您监控成本并支付账单的特征。	接收预算通知、价格变动通知和异常提示。有关更多信息，请参阅 AWS Billing 用户指南中的以下页面： • 为预算通知创建 Amazon SNS 主题 • 设置通知 • 检测不寻常的支出 AWS Cost Anomaly Detection

业务应用程序服务

下表介绍了 Amazon Chime 如何与亚马逊集成SNS以发送重要会议事件的通知，从而使您能够随时了解您的通信和日程安排。

您可以利用此集成来利用 Amazon Chime SDK 事件通知来增强组织内外的协作工具。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon Chime – 允许您在组织内外开会、聊天和拨打业务电话。	接收重要的会议事件通知。有关更多信息，请参阅《亚马逊 Chime 开发者指南》中的 A mazon Chime SDK 事件通知。

计算服务

下表描述了亚马逊如何与各种 AWS 计算服务SNS集成，使您能够接收有关关键事件的通知，例如自动缩放操作、EC2 Image Builder 完成、Elastic Beanstalk 环境变化、Lambda 函数输出和 Lightsail 指标阈值。

您可以利用这些集成，通过随时了解关键更新和操作来高效管理您的应用程序和资源。 AWS 服务

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon A EC2 uto Scaling — 帮助您获得正确数量的亚马逊弹性计算云 (AmazonEC2) 实例，用于处理应用程序的负载。	当 Auto Scaling 启动或终止您的 Auto Scaling 组中的亚马逊EC2实例时，会收到通知。有关更多信息，请参阅 Amazon Auto Scaling 用户指南中的 Auto Scaling 群组缩放时获取亚马逊SNSEC2通知。
EC2Image Builder — 帮助自动创建、管理和部署自定义、安全的 up-to-date服务器映像，这些映像已预先安装并预先配置了软件和设置，以满足特定 IT 标准。	在构建完成时接收通知。有关更多信息，请参阅 C AWS ompute 博客上的 “在 EC2 Image Builder 管道中跟踪最新服务器映像”。
AWS Elastic Beanstalk – 处理有关容量预配置、负载均衡、扩展您的应用程序和提供应用程序运行状况监控的部署细节。	接收影响您的应用程序的重要事件的通知。有关更多信息，请参阅开发者指南中的 Amazon AWS Elastic Beanstalk 的 Elastic Beanstalk 环境通知SNS。
AWS Lambda：您可以运行代码，而无需预置或管理服务器。	通过将SNS主题设置为 Lambda 死信队列或 Lambda 目标来接收函数输出数据。有关更多信息，请参阅 AWS Lambda 开发人员指南中的异步调用。
Amazon Lightsail — 帮助开发人员开始使用 AWS 来构建网站或网络应用程序。	在您的实例、数据库或负载均衡器的指标超过指定阈值时接收通知。有关更多信息，请参阅 Amazon Lightsail 开发人员指南中的在 Amazon Lightsail 中添加通知联系人。

容器服务

下表描述了亚马逊如何SNS与 Amazon EKS Distro 和 Amazon 等 AWS 容器服务集成ECS，允许您跟踪亚马逊EKS集群的更新和安全补丁，并接收有关ECS经过优化的AMI新版本的通知。

您可以随时了解重要的更新和更改，从而利用这些集成来维护容器部署的安全性和效率。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon EKS Distro — 无论您的应用程序部署在何处，都可让您创建可靠、安全的集群。	跟踪使用 Amazon EKS Distro 创建的集群的更新和安全补丁。有关更多信息，请参阅 Amazon EKS Distro 简介，这是亚马逊使用的开源 Kubernetes 发行版。EKS
亚马逊弹性容器服务 (AmazonECS) — 使您能够在集群上运行、停止和管理容器。	当新的 Amazon ECS 优化版本可用AMI时收到通知。有关更多信息，请参阅《亚马逊弹性容器服务开发者指南》中的订阅亚马逊ECS优化的AMI更新通知。

客户参与服务

下表描述了亚马逊如何通过与 Amazon Connect 和 Amazon Simple Email Service (SES) 集成来SNS增强客户参与服务，使您能够接收警报和验证、配置双向SMS消息以及监控退回、投诉和送达的电子邮件通知。 AWS End User Messaging SMS

这些集成可帮助您管理跨多个渠道的客户通信。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon Connect – 允许您设置全渠道云联络中心以与客户互动。	接收提示和验证。有关更多信息，请参阅《Amazon Connect 管理员指南》中的 Amazon Connect 的力量。 AWS
AWS End User Messaging SMS— 通过向客户发送电子邮件、语音消息SMS和推送通知，帮助您吸引他们。	配置双向模式SMS，允许您接收来自客户的消息。有关更多信息，请参阅《AWS End User Messaging SMS 用户指南》中的双向SMS消息。
Amazon Simple Email Service (AmazonSES) — 为您提供经济实惠的方式，让您使用自己的电子邮件地址和域名发送和接收电子邮件。	接收退回邮件、投诉和送达通知。有关更多信息，请参阅《亚马逊简单电子邮件服务开发者指南》SES中的为亚马逊配置亚马逊SNS通知。

数据库服务

下表描述了亚马逊如何SNS与 AWS AWS Database Migration Service (DMS)、Amazon DynamoDB、亚马逊、Amazon Neptune、 ElastiCache Amazon Redshift 和亚马逊关系数据库服务 () 等数据库服务集成，以发送有关数据迁移、维护活动、缓RDS存更新和数据库更改等重要事件的通知。

这些集成通过提供有关关键操作事件的及时警报，帮助您更有效地监控和管理数据库环境。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS Database Migration Service— 将数据从本地数据库迁移到。 AWS Cloud	AWS DMS 事件发生时接收通知；例如，创建或删除复制实例时。有关更多信息，请参阅 AWS Database Migration Service 用户指南中的在 AWS Database Migration Service中使用事件和通知。
Amazon DynamoDB — 在这款完全托管SQL的无数据库服务中，提供快速且可预测的性能以及无缝可扩展性。	在发生维护事件时接收通知。有关更多信息，请参阅 Amazon DynamoDB 开发者指南中的自定义DAX集群设置。
Amazon ElastiCache — 提供高性能、可调整大小且经济实惠的内存缓存，同时消除与部署和管理分布式缓存环境相关的复杂性。	在发生重要事件时接收通知。有关更多信息，请参阅亚马逊 ElastiCache (Memcached) 用户指南SNS中的事件通知和亚马逊。
Amazon Neptune – 允许您构建和运行与高度连接的数据集配合使用的应用程序。	在 Neptune 事件发生时接收通知。有关更多信息，请参阅 Neptune 用户指南中的使用 Neptune 事件通知。
Amazon Redshift – 管理数据仓库的所有设置、操作和扩展工作。	接收 Amazon Redshift 事件的通知。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 事件通知。
Amazon Relational Database Servic e — 可以更轻松地在中设置、操作和扩展关系数据库 AWS Cloud。	接收有关亚马逊RDS事件的通知。有关更多信息，请参阅亚马逊RDS用户指南中的使用亚马逊RDS事件通知。

开发人员工具服务

下表描述了 Amazon 如何与 AWS 开发者工具服务（例如、 AWS CodeBuild、、 AWS CodeCommit AWS CodeDeploy CodeGuru AWS CodePipeline、Amazon 和）SNS集成 AWS CodeStar，以针对关键事件（例如构建状态更改、存储库更新、部署进度、性能异常和管道操作）提供通知。

这些集成通过及时接收有关重要事件的警报，帮助您有效地监控和管理软件开发工作流程。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS CodeBuild – 可编译源代码，运行单元测试，并生成可供部署的项目。	在构建成功、失败或从一个构建阶段迁移到另一个构建阶段时接收通知。有关更多信息，请参阅《AWS CodeBuild 用户指南》 CodeBuild中的生成通知示例。
AWS CodeCommit – 提供版本控制，以在云中私有存储和管理资产。	接收有关 CodeCommit 仓库事件的通知。有关更多信息，请参阅AWS CodeCommit 用户指南中的示例：为 Amazon SNS 主题创建 AWS CodeCommit 触发器。
AWS CodeDeploy— 自动将应用程序部署到亚马逊EC2实例、本地实例、无服务器 Lambda 函数或亚马逊服务。ECS	接收有关 CodeDeploy 部署或实例事件的通知。有关更多信息，请参阅《AWS CodeDeploy 用户指南》中的为 CodeDeploy 事件创建触发器。
Amazon CodeGuru — 从您的实时应用程序收集运行时性能数据，并提供建议，以帮助您微调应用程序性能。	在发生异常时接收通知。有关更多信息，请参阅A mazon CodeGuru 用户指南中的处理异常和建议报告。
AWS CodePipeline – 自动执行持续发布软件更改所需的步骤。	接收有关批准操作的通知。有关更多信息，请参阅《AWS CodePipeline 用户指南》 CodePipeline中的 “管理批准操作”。
AWS CodeStar – 在 AWS上创建、管理和处理软件开发项目。	接收有关您使用的资源中所发生事件的通知。有关更多信息，请参阅《开发者工具控制台用户指南》中的配置 Amazon 通知SNS主题。

前端 Web 和移动服务

下表描述了 Ama SNS zon 如何通过发送电子邮件SMS、语音消息和推送通知（包括配置双向SMS接收客户消息的功能）来增强客户参与度。 AWS End User Messaging SMS

这种集成使您能够通过各种沟通渠道更有效地与客户互动。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS End User Messaging SMS— 通过向客户发送电子邮件、语音消息SMS和推送通知，帮助您吸引他们。	配置双向模式SMS，允许您接收来自客户的消息。有关更多信息，请参阅《AWS End User Messaging SMS 用户指南》中的双向SMS消息。

游戏开发服务

下表描述了亚马逊如何与亚马逊SNS集成， GameLift 为基于会话的多人游戏服务器中的配对和队列事件提供通知。

这种集成可帮助游戏开发者自动执行和监控其游戏服务器的部署、操作和扩展，从而确保无缝的游戏体验。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon GameLift — 为在云端托管基于会话的多人游戏服务器提供解决方案，包括用于部署、操作和扩展游戏服务器的完全托管服务。	接收匹配和队列事件通知。有关更多信息，请参阅以下页面： • 有关配对通知，请参阅《Amazon GameLift FlexMatch 开发者指南》中的设置 FlexMatch 事件通知。 • 有关队列通知，请参阅 Amazon GameLift 开发者指南中的为游戏会话放置设置事件通知。

物联网服务

下表描述了 Amazon 如何与、、和 AWS IoT Greengrass等 AWS IoT AWS IoT Core服务SNS集成 AWS IoT Device Defender AWS IoT Events，为物联网事件和警报提供通知。

这些集成使您可以有效地监控设备行为，接收异常活动的警报，并通过实时更新和操作来管理物联网设备。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS IoT Core— 提供将您的物联网设备连接到其他设备和服务的云 AWS Cloud 服务。	接收 AWS IoT Core 事件通知。有关更多信息，请参阅AWS IoT 开发者指南中的创建 Amazon SNS 规则。
AWS IoT Device Defender – 审核设备的配置，监控互联设备以检测异常行为，并降低安全风险。	在设备违反行为时接收告警。有关更多信息，请参阅《AWS IoT 开发者指南》中的如何使用 d AWS IoT Device Defender etec t。
AWS IoT Events – 让您了解如何监控您的设备和设备机群中的故障情况或操作中的更改，并在发生此类事件时触发措施。	接收 AWS IoT Events 事件通知。有关更多信息，请参阅 AWS IoT Events 开发人员指南中的 Amazon Simple Notification Service。
AWS IoT Greengrass— 扩展 AWS 到物理设备，这样它们就可以根据自己生成的数据在本地采取行动，同时仍然使用云进行管理、分析和持久存储。	接收 AWS IoT Greengrass 事件通知。有关更多信息，请参阅《AWS IoT Greengrass Version 1 开发人员指南》中的SNS连接器。

机器学习服务

下表描述了亚马逊如何与 AWS 机器学习服务（例如亚马逊、Amazon DevOps Guru CodeGuru、Amazon Lookout for Metrics、Amazon Rekognition和 SageMaker亚马逊）SNS集成，以提供异常通知、运营见解和数据标签活动。

这些集成允许您监控应用程序性能，接收数据异常警报，并通过实时更新简化机器学习模型的部署。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon CodeGuru — 从您的实时应用程序收集运行时性能数据，并提供建议，以帮助您微调应用程序性能。	在发生异常时接收通知。有关更多信息，请参阅 A mazon CodeGuru 用户指南中的处理异常和建议报告。
Amazon DevOps Guru — 使用机器学习生成运营见解，以帮助您提高运营应用程序的性能。	转发洞察和确认。有关更多信息，请参阅AWS 管理与治理博客上的 “通过 PagerDuty Amazon DevOps Guru 向待命团队提供基于机器学习的运营见解”。
Amazon Lookout for Metrics – 查找数据中的异常情况，确定其根本原因，并使您能够快速采取措施。	接收异常通知。有关更多信息，请参阅《亚马逊 SNS Lookout for Metrics 开发者指南》中的使用亚马逊和 Lookout for Metrics。
Amazon Rekognition – 让您能够将图像和视频分析添加到您的应用程序	接收请求结果通知。有关更多信息，请参阅 Amazon Rekognition 开发人员指南中的参考：视频分析结果通知。
Amazon SageMaker — 使数据科学家和开发人员能够构建和训练机器学习模型，然后将其直接部署到可用于生产的托管环境中。	在标记数据对象时接收通知。有关更多信息，请参阅《Amazon SageMaker 开发者指南》中的创建流式标签任务。

管理和治理服务

下表描述了 Amazon 如何与 AWS 管理和治理服务（例如 AWS Chatbot、、、 AWS CloudFormation、 CloudTrail、、 CloudWatch、 AWS Config AWS Control Tower AWS License Manager AWS Service Catalog、和）SNS集成，为基础设施变更 AWS Systems Manager、合规性警报和运营见解等关键事件提供通知。

这些集成通过向相关团队和系统提供及时的警报和更新，帮助您高效地监控和管理 AWS 环境。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS Chatbot— 使 DevOps 软件开发团队能够使用 Amazon Chime 和 Slack 聊天室来监控和响应中的运营事件。 AWS Cloud	将通知发送到聊天室。有关更多信息，请参阅 AWS Chatbot 管理员指南中的设置 AWS Chatbot。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS CloudFormation— 使您能够以可预测的方式重复创建和配置 AWS 基础架构部署。	在创建和更新堆栈时接收通知。有关更多信息，请参阅 AWS CloudFormation 用户指南中的设置 AWS CloudFormation 堆栈选项。
AWS CloudTrail – 提供您的 AWS 账户 活动的事件历史记录。	将新的日志文件 CloudTrail 发布到您的 Amazon S3 存储桶时会收到通知。有关更多信息，请参阅AWS CloudTrail 用户指南 CloudTrail中的配置 Amazon SNS 通知。
Amazon CloudWatch — 实时监控您的 AWS 资源和您运行 AWS 的应用程序。	在告警状态更改时接收通知。有关更多信息，请参阅亚马逊 CloudWatch 用户指南中的使用亚马逊 CloudWatch 警报。
AWS Config— 提供中 AWS 资源配置的详细视图 AWS 账户。	在更新资源时，或者在 AWS Config 针对您的资源评估自定义规则或托管规则时接收通知。有关更多信息，请参阅《AWS Config 开发人员指南》中的AWS Config 发送到SNS主题的通知和示例配置项目变更通知。
AWS Control Tower— 使您能够设置和管理安全、合规的多账户 AWS 环境。	使用提示可帮助您防止登录区内的漂移，并接收合规性通知。有关更多信息，请参阅 AWS Control Tower 用户指南中的通过 Amazon Simple Notification Service 跟踪提示。
AWS License Manager— 帮助您在本地环境中 AWS 集中管理软件供应商提供的软件许可证。	接收 License Manager 通知和提示。有关更多信息，请参阅《L icense Manager 用户指南》中的 License Manager 设置和 “ AWS 管理和治理” 博客上的 “创建 AWS License Manager 通知 ServiceNow 事件”。
AWS Service Catalog – 使 IT 管理员可以创建、管理和向最终用户分发已批准的产品组合，然后，最终用户可以在个性化的门户中访问他们所需的产品。	接收有关堆栈事件的通知。有关更多信息，请参阅《Service Catalog 管理员指南》中的 AWS Service Catalog 通知约束。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS Systems Manager— 允许您在上查看和控制您的基础架构 AWS。	接收有关命令状态的通知。有关更多信息，请参阅AWS Systems Manager 用户指南中的使用亚马逊SNS通知监控 Systems Manager 的状态变化。

媒体服务

下表描述了亚马逊如何与 Amazon Elastic Transcoder SNS 集成，以便在媒体转码任务状态发生变化时发送通知，从而使您能够有效地监控和管理存储在 Amazon S3 中的媒体文件转换为适合消费者播放设备的格式的过程。

这种集成通过提供有关作业状态的实时警报，帮助您简化媒体处理工作流程。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon Elastic Transcoder – 让您可以将 Amazon S3 中存储的媒体文件转换为消费者播放设备所要求的媒体文件格式。	在作业状态更改时接收通知。有关更多信息，请参阅 Amazon Elastic Transcoder 开发人员指南中的任务状态通知。

迁移和传输服务

下表介绍了 Amazon 如何与 AWS 迁移和传输服务（例如 AWS Application Discovery Service、 AWS Database Migration Service (DMS) 和 ）集SNS成 AWS Snowball，以提供服务器数据收集、数据库迁移活动和数据传输任务等事件的通知。

这些集成通过提供有关关键迁移任务的实时警报和更新，帮助您有效地管理和监控云迁移流程。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS Application Discovery Service— AWS Cloud 通过收集有关本地服务器的使用情况和配置数据，帮助您规划向的迁移。	通过接收事件通知 AWS CloudTrail。有关更多信息，请参阅《Applicati on Discovery Service 用户指南》 AWS CloudTrail中的使用记录应用程序发现服务API调用。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS Database Migration Service— 将数据从本地数据库迁移到。 AWS Cloud	AWS DMS 事件发生时接收通知；例如，创建或删除复制实例时。有关更多信息，请参阅 AWS Database Migration Service 用户指南中的在 AWS Database Migration Service中使用事件和通知。
AWS Snowball— 使用物理存储设备在 Amazon S3 和您的现场数据存储位置之间 faster-than-internet 快速传输大量数据。	接收 Snowball 作业的通知。有关更多信息，请参阅下列内容： • AWS Snowball 用户指南中的 Snowball 通知 • 第 5 步：在《AWS Snowball Edge 开发者指南》中选择您的通知首选项 • 第 5 步：在《AWS Snowcone 用户指南》中选择您的通知首选项

网络和内容分发服务

下表描述了亚马逊如何与 AWS 网络和内容分发服务（例如亚马逊API网关、亚马逊、Elastic Load Balancing CloudFront AWS Direct Connect、Amazon Load Balancing、Amazon Route 53 和亚马逊 VPC）SNS集成，以发送API消息、 CloudFront 指标警报、连接状态更改、负载均衡器事件、运行状况检查状态和VPC终端节点活动等事件的通知。

这些集成通过提供及时的警报和更新，帮助您监控和管理您的网络和内容交付操作。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon API Gatew ay — 使您能够以任何规模创建REST和 WebSocket APIs部署自己的网关。	接收发布到API网关终端节点的消息。有关更多信息，请参阅《API网关开发者指南》中的教程：RESTAPI使用 AWS 集成功能构建API网关。

AWS 服务	在 Amazon 上使用的好处 SNS
亚马逊 CloudFront — 加快静态和动态网页内容的分发，例如.html、.css、.php、图像和媒体文件。	当基于指定 CloudFront 指标的警报发生时接收通知。有关更多信息，请参阅《Amazon CloudFront 开发者指南》中的设置警报以接收通知。
AWS Direct Connect— 通过标准以太网光纤电缆将您的内部网络链接到某个 AWS Direct Connect 位置。	在监控 AWS Direct Connect 连接状态的告警状态更改时接收通知。有关更多信息，请参阅《AWS Direct Connect 用户指南》中的创建 CloudWatch 警报以监控 AWS Direct Connect 连接。
Elastic Load Balancing — 自动将您的传入流量分配到多个或多个可用区域中的多个目标，例如亚马逊EC2实例、容器和 IP 地址。	接收您为负载均衡器事件创建的告警的通知。有关更多信息，请参阅《传统负载均衡器用户指南》中的为负载均衡器创建 CloudWatch 警报。
Amazon Route 53 — 提供域名注册、DNS路由和运行状况检查。	在运行状况检查状态为运行不佳时接收通知。有关更多信息，请参阅 Amazon R oute 53 开发者指南中的在运行状况检查状态为不健康时接收亚马逊SNS通知（控制台）。
Amazon Virtual Private Cloud（亚马逊VPC）— 使您能够将 AWS 资源启动到您定义的虚拟网络中。	接收接口终端节点上发生的特定事件的提醒。有关更多信息，请参阅 Amazon VPC 用户指南中的创建和管理终端节点服务的通知。

安全性、身份与合规性服务

下表描述了亚马逊如何与 AWS 安全、身份和合规服务（例如 AWS Directory Service亚马逊 GuardDuty、Amazon Inspector）SNS集成 AWS Security Hub，并提供有关目录状态更改、安全发现、Inspector 事件和安全中心公告的通知。

这些集成通过提供有关安全和合规事件的及时警报和更新，帮助您保持稳健的安全实践。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS Directory Service— 提供多种与其他方式一起使用 Microsoft 活动目录 (AD) 的方法 AWS 服务。	当您的目录状态发生变化时，接收电子邮件或短信 (SMS)。有关更多信息，请参阅 AWS Directory Service 管理指南中的配置目录状态通知。
Amazon GuardDuty — 提供持续的安全监控，以帮助识别 AWS 环境中意外的、可能未经授权的或恶意的活动。	接收有关新发布的调查结果类型、现有调查结果类型的更新以及其他功能更改的通知的信息。有关更多信息，请参阅 Amazon GuardDuty 用户指南中的订阅 GuardDuty 公告SNS主题。
Amazon Inspector — 测试您的亚马逊EC2实例的网络可访问性以及在这些实例上运行的应用程序的安全状态。	接收有关 Amazon Inspector 事件的通知。有关更多信息，请参阅 Amazon In spector 用户指南中的为亚马逊检查员通知设置SNS主题。
AWS Security Hub – 自动执行 AWS 安全检查并集中管理安全提示。	接收有关 AWS Security Hub 公告的通知，包括有关已添加、编辑或停用的 AWS Security Hub 控件或标准的通知。有关更多信息，请参阅通过 Amazon SNS 订阅 AWS Security Hub 公告。

无服务器服务

下表描述了亚马逊如何与亚马逊 DynamoDB、 EventBridge Amazon 和 Lambda 等服务SNS集成，以发送维护更新、实时数据流和 Lambda 函数输出等关键事件的通知。

这些集成通过及时接收有关关键操作的警报并自动响应这些事件，帮助您高效地监控和管理应用程序。

AWS 服务	在 Amazon 上使用的好处 SNS
Amazon DynamoDB — 在这款完全托管SQL的无数据库服务中，提供快速且可预测的性能以及无缝可扩展性。	在发生维护事件时接收通知。有关更多信息，请参阅 Amazon DynamoDB 开发者指南中的自定义DAX集群设置。
亚马逊 EventBridge — 提供来自您自己的应用程序、 software-as-a-service (SaaS) 应用程序的实时数据流， AWS 服务 并将这些数据传送到	接收 EventBridge 事件通知。有关更多信息，请参阅《亚马逊 EventBridge 用户指南》中的亚马逊 EventBridge目标。

AWS 服务	在 Amazon 上使用的好处 SNS
目标，包括亚马逊SNS。 EventBridge 以前叫做“ CloudWatch 活动”。	
AWS Lambda：您可以运行代码，而无需预置或管理服务器。	通过将SNS主题设置为 Lambda 死信队列或 Lambda 目标来接收函数输出数据。有关更多信息，请参阅 AWS Lambda 开发人员指南中的异步调用。

存储服务

下表描述了亚马逊如何SNS与 Amazon Elastic File System (EFS)、Amazon S3 Glacier、Amazon S3 等 AWS 存储服务集成，以及 AWS Snowball 如何针对备份活动、文件系统警报、数据检索任务、存储桶更改和数据传输操作等各种事件提供通知。 AWS Backup

这些集成通过及时接收有关关键存储事件的警报，帮助您高效地监控和管理存储解决方案。

AWS 服务	在 Amazon 上使用的好处 SNS
AWS Backup— 帮助您在云端和本地集中和自动备份数据 AWS 服务	接收 AWS Backup 事件通知。有关更多信息，请参阅《AWS Backup 开发者指南》中的使用 Amazon SNS 跟踪 AWS Backup 事件。
Amazon Elastic File System — 为您的亚马逊EC2实例提供文件存储。	接收有关您为 Amazon EFS 事件创建的警报的通知。有关更多信息，请参阅 Amazon Elastic File System 用户指南中的自动监控工具。
Amazon S3 Glacier – 为不经常使用的数据提供存储。	在文件库上设置通知配置，以便在任务完成时向SNS主题发送消息。有关更多信息，请参阅 Amazon S3 Glacier 开发人员指南中的在 Amazon S3 Glacier 中配置文件库通知。
Amazon Simple Storage Service (Amazon S3) – 提供对象存储服务。	在 Amazon S3 存储桶发生更改时或在对象未复制到目标区域的罕见情况下接收通知。有关更多信息，请参阅《亚马逊简单存储服务用户指南》中的演练：为通知（SNS主题或SQS队列）配置

AWS 服务	在 Amazon 上使用的好处 SNS
	存储桶以及使用复制指标和 Amaz on S3 事件通知监控进度。
AWS Snowball— 使用物理存储设备在 Amazon S3 和您的现场数据存储位置之间 faster-than-internet 快速传输大量数据。	接收 Snowball 作业的通知。有关更多信息，请参阅下列内容： • AWS Snowball 用户指南中的 Snowball 通知 • 第 5 步：在《AWS Snowball Edge 开发者指南》中选择您的通知首选项 • 第 5 步：在《AWS Snowcone 用户指南》中选择您的通知首选项

其他事件来源

下表描述了SNS如何使用 Amazon 及时接收有关 AWS 每日功能更新和 AWS IP 地址范围变化的通知，确保您了解最新的 AWS 服务版本、实例类型、VPC终端节点和公有 IP 地址的变化。

这些集成可帮助您顺应基础 AWS 架构 up-to-date 的变化并有效地管理您的资源。

来源	在 Amazon 上使用的好处 SNS
AWS 每日功能更新	AWS 通过 Amazon SNS 主题及时接收有关版本和更新的详细信息。这些版本包括 AWS 区域、 AWS 服务、与 AWS 服务配额 AWS 服务 集成的亚马逊VPC终端节点、亚马逊实例类型、亚马逊EC2实例类型、Amazon SageMaker Nimble Studio 实例类型、亚马逊RDS数据库引擎版本和亚马逊 A MSK pache Kafka 版本。有关更多信息，请参阅AWS 新闻博客SNS中的通过 Amazon 订阅 AWS 每日专题更新。
AWS IP 地址范围	通过 Amazon SNS 主题接收有关 AWS IP 范围变更的通知。有关更多信息，请参阅中的 AWS

来源	在 Amazon 上使用的好处 SNS
	IP 地址范围通知 Amazon Web Services 一般参考，以及AWS 新闻博客中的通过 Amazon SNS 订阅 AWS 公有 IP 地址变更。

有关事件驱动型计算的更多信息，请参阅以下资源：

- 什么是事件驱动型架构？
- Compute 博客上的 Amazon 事件驱动 AWS 计算SNS以及计算、存储、数据库和网络服务AWS
- 在 Compute 博客上使用 AWS 事件分叉管道丰富事件驱动架构AWS

亚马逊SNS活动目的地

此页面按照 application-to-application (A2A) 消息和 application-to-person (A2P) 通知分组列出了可以接收事件信息的所有目的地。

> **Note**
>
> 亚马逊在 2020 年 10 月SNS推出了FIFO话题。当前，大多数 AWS 服务仅支持接收来自SNS标准主题的事件。Amazon SQS 支持接收来自SNS标准和FIFO主题的事件。

主题

- A2A 目标
- A2P 目标

A2A 目标

下表描述了亚马逊SNS如何将事件传送到各个 application-to-application (A2A) 目的地，例如亚马逊 Data Firehose、Lambda、Amazon、Event Fork SQS Pipelines 和 AWS /S 终端节点。HTTP

这些集成允许您存档和分析数据、触发自定义业务逻辑、促进应用程序集成以及将事件路由到外部 Webhook，从而提高事件驱动架构的效率和灵活性。

事件目标	在 Amazon 上使用的好处 SNS
Amazon Data Firehose	将事件传输到传输流以进行存档和分析。通过传输流，您可以将事件传送到亚马逊简单存储服务 (Amazon S3)、Amazon Redshift 和 OpenSearch 亚马逊服务（OpenSearch 服务）等 AWS 目的地，或第三方目的地，例如Datadog、New Relic、MongoDB和Splunk。有关更多信息，请参阅 Fanout Amazon SNS 通知，带有 Firehose 传送流，可增强数据管理。
AWS Lambda	将事件传输到函数，用于触发自定义业务逻辑的执行。有关更多信息，请参阅 Fanout Amazon SNS 向 Lambda 函数发送通知，用于自动处理。
Amazon SQS	将事件传输到队列，以进行应用程序集成。有关更多信息，请参阅 Fanout Amazon SNS 通知到亚马逊SQS队列进行异步处理。
AWS 事件分叉管道	将事件传输到事件备份和存储、事件搜索和分析或事件重放管道。有关更多信息，请参阅 Fanout Amazon SNS 事件到 AWS 事件分叉管道。
HTTP/S	向外部 Webhook 传输事件。有关更多信息，请参阅 Fanout Amazon SNS 向 HTTP /S 端点发送通知。

A2P 目标

下表描述了亚马逊如何向各种目的地发送 application-to-person (A2P) 通知，包括通过手机SMS和原生推送通知、电子邮件收件箱、Amazon Chime 聊天室、Slack 频道以及通过以下方式向待命团队SNS提供运营见解。 PagerDuty

这些集成通过支持跨多个平台和通信渠道的实时警报和更新，提高了通信和运营效率。

事件目标	在 Amazon 上使用的好处 SNS
SMS	以短信形式将事件传输到移动电话。有关更多信息，请参阅 通过 Amazon 发送移动短信 SNS。
Email	将事件作为电子邮件发送到收件箱。有关更多信息，请参阅 Amazon SNS 电子邮件订阅设置和管理。
平台终端节点	将事件作为本机推送通知发送到移动电话。有关更多信息，请参阅 通过 Amazon 发送移动推送通知 SNS。
AWS Chatbot	将事件传输到 Amazon Chime 聊天室或 Slack 频道。有关更多信息，请参阅 AWS Chatbot 管理指南中的以下页面。 • AWS Chatbot 使用 Amazon Chime 进行设置 • AWS Chatbot 使用 Slack 进行设置 • 与其他 AWS 服务 AWS Chatbot 一起使用
PagerDuty	向待命团队传输运营洞察。有关更多信息，请参阅AWS 管理与治理博客上的 “通过 PagerDuty Amazon DevOps Guru 向待命团队提供基于机器学习的运营见解”。

Note

您可以将原生 AWS 事件和自定义事件发送到聊天应用程序：

- 原生 AWS 事件 — 您可以使用通过亚马逊SNS主题 AWS Chatbot 向 Amazon Chime 和 Slack 发送原生 AWS 事件。支持的原生 AWS 事件集包括来自 AWS Billing and Cost Management、、 AWS Health AWS CloudFormation CloudWatch、Amazon 等的事件。有关更多信息，请参阅《AWS Chatbot 管理员指南》中的AWS Chatbot 与其他服务一起使用。

- 自定义事件 — 您还可以通过亚马逊SNS主题将自定义事件发送给 Amazon Chime、Slack 和 Microsoft Teams。为此，您需要向主题发布自定义事件，该SNS主题将事件传送到已订阅的 Lambda 函数。然后，Lambda 函数使用聊天应用程序的 Webhook 将事件传输给收件人。有关更多信息，请参阅如何使用 webhook 将亚马逊SNS消息发布到 Amazon Chime、Slack 或 Microsoft Team s？

使用 Amazon SNS 发送 application-to-application A2A 消息

本主题讨论在处理 application-to-application (A2A) 消息和各种扇出场景（例如向 Firehose、Lambda 函数、Amazon 队列和 /S 终端节点传送消息）SNS时如何有效地记录和监控亚马逊SQS。HTTP您将探索使用记录亚马逊SNSAPI呼叫如何 CloudTrail 帮助跟踪与您的亚马逊SNS主题的互动，例如订阅传输流或终端节点，以及管理消息传送。这种日志记录对于审计和确保您的邮件基础设施保持安全并符合您的组织策略至关重要。

本主题还介绍如何使用监控亚马逊SNS主题 CloudWatch，您将在其中设置指标来跟踪跨多个目的地（例如 Amazon S3、Amazon Redshift 和服务）的分组通知的表现。 OpenSearch 了解如何配置 CloudWatch 警报以通知您任何问题，例如交付失败或延迟增加，从而使您能够及时做出响应。

通过记录和监控您的 Amazon SNS 主题，您可以确保消息分散场景顺利运行，并且可以主动解决可能出现的任何问题，从而保持消息传递工作流程的完整性。

主题

- Fanout Amazon SNS 通知，带有 Firehose 传送流，可增强数据管理
- Fanout Amazon SNS 向 Lambda 函数发送通知，用于自动处理
- Fanout Amazon SNS 通知到亚马逊SQS队列进行异步处理
- Fanout Amazon SNS 向 HTTP /S 端点发送通知
- Fanout Amazon SNS 事件到 AWS 事件分叉管道
- 在亚马逊上使用 Amazon EventBridge 排程器 SNS

Fanout Amazon SNS 通知，带有 Firehose 传送流，可增强数据管理

您可以将 Amazon Data Firehose 传输流订阅到亚马逊SNS主题，这样您就可以向其他存储和分析终端节点发送通知。发布到亚马逊SNS主题的消息将发送到已订阅的 Firehose 传送流，然后按照 Firehose 中的配置发送到目的地。订阅所有者最多可以为一个亚马逊SNS主题订阅五个 Firehose 直播流。每个 Firehose 传输流都有默认的请求配额和每秒吞吐量。此限制可能会导致发布的消息（入站流量）多于传输的消息（出站流量）。当入站流量多于出站流量时，您的订阅可能会累积大量的消息积压，从而可能会导致较长的消息传输延迟。您可以根据发布率请求增加限额，以避免对您的工作负载产生不利影响。

通过Firehose传送流，你可以将亚马逊的SNS通知分散到亚马逊简单存储服务（Amazon S3）、亚马逊Redshift、亚马逊服务（服务）以及第三方OpenSearch 服务提供商， OpenSearch 例如Datadog、New Relic、MongoDB和Splunk。

例如，您可以使用此功能将发送到 Amazon S3 存储桶中的主题的消息永久存储以用于合规性、存档或其他目的。为此，请创建带有 S3 存储桶目标的 Firehose 传输流，并将该传输流订阅亚马逊SNS主题。再举一个例子，要对发送到 Amazon SNS 主题的消息进行分析，请创建一个带有 OpenSearch 服务索引目标的传输流。然后，您可以将 Firehose 直播订阅亚马逊SNS主题。

亚马逊SNS还支持记录发送到 Firehose 终端节点的通知的消息传输状态。有关更多信息，请参阅 Amazon SNS 消息传送状态。

主题

- 将 Firehose 直播订阅亚马逊主题的先决条件 SNS
- 将 Firehose 直播订阅亚马逊话题 SNS
- 管理跨多个传送流目的地的 Amazon SNS 消息
- Amazon SNS 消息存档和分析：航空票务平台的示例用例

将 Firehose 直播订阅亚马逊主题的先决条件 SNS

要将 Amazon Data Firehose 传送流订阅某个SNS主题，您 AWS 账户 必须具备以下条件：

- 标准SNS话题。有关更多信息，请参阅 创建亚马逊SNS主题。
- Firehose 的直播流。有关更多信息，请参阅《亚马逊数据 Firehose 开发者指南》中的创建亚马逊数据 Firehose 传输流并向您的应用程序授予对您的 Firehos e 资源的访问权限。
- 信任 Amazon SNS 服务主体并有权写入传输流的 AWS Identity and Access Management (IAM) 角色。您将在创建订阅`SubscriptionRoleARN`时输入此角色的 Amazon 资源名称 (ARN)。亚马逊SNS担任这个角色，这使亚马逊SNS能够将记录放入Firehose的交付流中。

  下面的示例策略显示了建议的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:DescribeDeliveryStream",
        "firehose:ListDeliveryStreams",
```

```
        "firehose:ListTagsForDeliveryStream",
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Resource": [
        "arn:aws:firehose:us-east-1:111111111111:deliverystream/firehose-sns-
delivery-stream"
      ],
      "Effect": "Allow"
    }
  ]
}
```

要提供使用 Firehose 的完全权限，您还可以使用 AWS 托管策略。`AmazonKinesisFirehoseFullAccess`或者，要为使用 Firehose 提供更严格的权限，您可以创建自己的策略。至少，策略必须提供在特定传输流上运行 `PutRecord` 操作的权限。

在所有情况下，您还必须编辑信任关系以包括 Amazon SNS 服务委托人。例如：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

有关创建角色的更多信息，请参阅《IAM用户指南》中的创建角色以向 AWS 服务委派权限。

完成这些要求后，您可以通过直播订阅该SNS主题。

将 Firehose 直播订阅亚马逊话题 SNS

要向 Amazon Data Firehose 传送流发送亚马逊SNS通知，请首先确保您已满足所有先决条件。有关支持的终端节点列表，请参阅中的 Amazon Data Firehose 终端节点和配额。Amazon Web Services 一般参考

向 Firehose 直播订阅某个主题

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择订阅。
3. 在 Subscriptions（订阅）页面上，选择 Create subscription（创建订阅）。
4. 在 Create subscription（创建订阅）页上的 Details（详细信息）部分中，执行以下操作：
   a. 对于主题 ARN，选择标准主题的 Amazon 资源名称 (ARN)。
   b. 对于协议，请选择 Firehose。
   c. 对于 Endpoint，选择可以接收来自亚马逊ARN的通知的 Firehose 传输流。SNS
   d. 对于订阅角色 ARN，请指定您为写入 Firehose 交付流而创建的 AWS Identity and Access Management (IAM) 角色。ARN有关更多信息，请参阅 将 Firehose 直播订阅亚马逊主题的先决条件 SNS。
   e. （可选）要从已发布的消息中移除任何 Amazon SNS 元数据，请选择启用原始消息传送。有关更多信息，请参阅 Amazon SNS 原始消息传送。
5. （可选）要配置筛选策略，请展开 Subscription filter policy（订阅筛选策略）部分。有关更多信息，请参阅 亚马逊SNS订阅筛选政策。
6. （可选）要为订阅配置死信队列，请展开 Redrive policy (dead-letter queue)（重新驱动策略（死信队列））部分。有关更多信息，请参阅 Amazon SNS 死信队列。
7. 选择创建订阅。

控制台将创建订阅并打开订阅的 Details（详细信息）页面。

管理跨多个传送流目的地的 Amazon SNS 消息

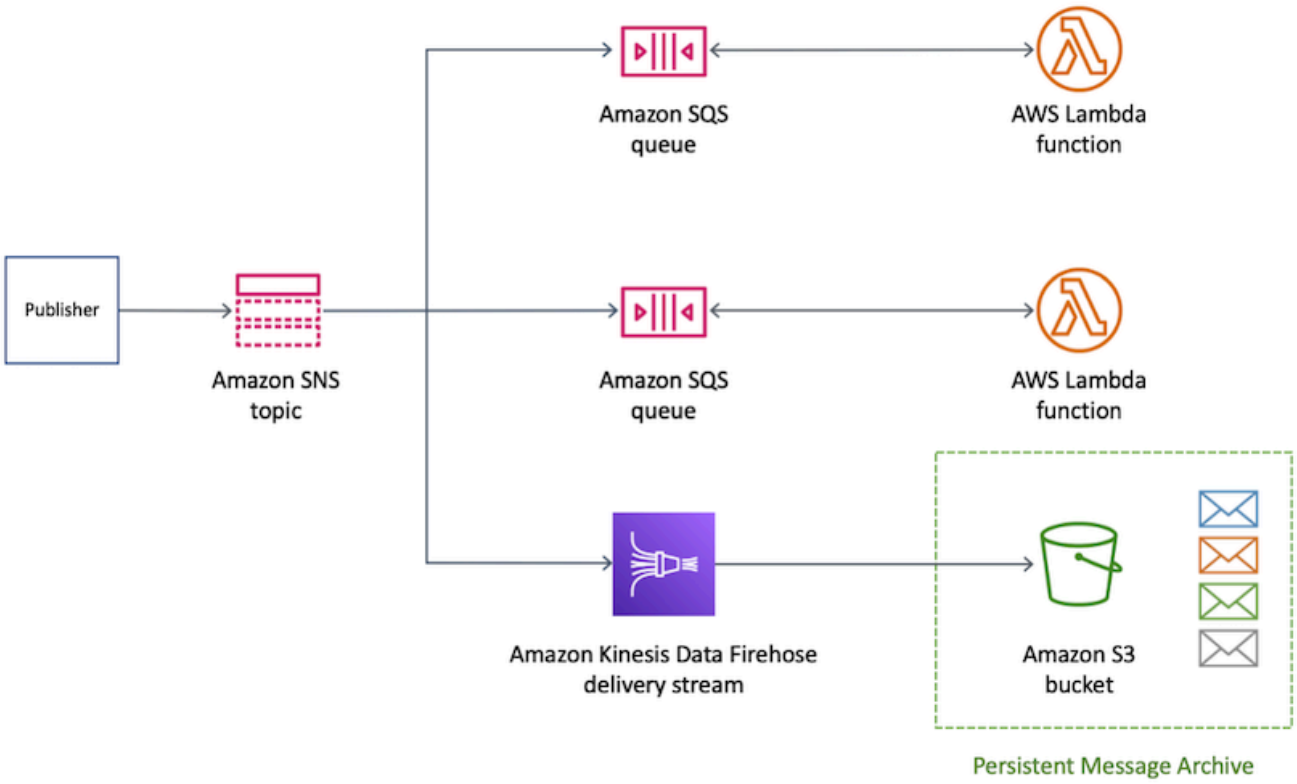
通过 Amazon Data Firehose 传输流，您可以向其他终端节点发送消息。本部分介绍如何使用受支持的目标。

主题

- 在 Amazon S3 目的地存储和分析亚马逊SNS消息
- 将亚马逊SNS消息与亚马逊 OpenSearch 服务目标集成
- 在亚马逊 Redshift 目的地配置亚马逊SNS消息传输和分析
- 使用 Amazon Data Firehose 配置亚马逊向HTTP目的地SNS发送消息

在 Amazon S3 目的地存储和分析亚马逊SNS消息

本节提供有关将数据发布到亚马逊简单存储服务 (Amazon S3) 的 Amazon Data Firehose 传输流的信息。



主题

- 格式化亚马逊SNS通知以存储在亚马逊 S3 目的地
- 使用 Athena 分析存储在 Amazon S3 中的亚马逊SNS消息

格式化亚马逊SNS通知以存储在亚马逊 S3 目的地

以下示例显示了发送到亚马逊简单存储服务 (Amazon S3) 存储桶的亚马逊SNS通知，该通知使用缩进来提高可读性。

> **Note**
>
> 在此示例中，已发布消息的原始消息传输被禁用。禁用原始消息传送后，Amazon SNS 会向消息添加JSON元数据，包括以下属性：
>
> - Type

- MessageId
- TopicArn
- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

有关原始消息传输的更多信息，请参阅 Amazon SNS 原始消息传送。

```
{
    "Type": "Notification",
    "MessageId": "719a6bbf-f51b-5320-920f-3385b5e9aa56",
    "TopicArn": "arn:aws:sns:us-east-1:333333333333:my-kinesis-test-topic",
    "Subject": "My 1st subject",
    "Message": "My 1st body",
    "Timestamp": "2020-11-26T23:48:02.032Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:333333333333:my-kinesis-test-
topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5",
    "MessageAttributes": {
        "myKey1": {
            "Type": "String",
            "Value": "myValue1"
        },
        "myKey2": {
            "Type": "String",
            "Value": "myValue2"
        }
    }
 }
```

以下示例显示了通过 Amazon Data Firehose 传输流SNS发送到同一 Amazon S3 存储桶的三条消息。将缓冲考虑在内，并且换行符分隔消息。

```
{"Type":"Notification","MessageId":"d7d2513e-6126-5d77-
bbe2-09042bd0a03a","TopicArn":"arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic","Subject":"My 1st subject","Message":"My 1st
 body","Timestamp":"2020-11-27T00:30:46.100Z","UnsubscribeURL":"https://
```

```
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
b59b-3b4aa6d8fcf5","MessageAttributes":{"myKey1":
{"Type":"String","Value":"myValue1"},"myKey2":{"Type":"String","Value":"myValue2"}}}
{"Type":"Notification","MessageId":"0c0696ab-7733-5bfb-b6db-
ce913c294d56","TopicArn":"arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic","Subject":"My 2nd subject","Message":"My 2nd
 body","Timestamp":"2020-11-27T00:31:22.151Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
b59b-3b4aa6d8fcf5","MessageAttributes":{"myKey1":{"Type":"String","Value":"myValue1"}}}
{"Type":"Notification","MessageId":"816cd54d-8cfa-58ad-91c9-8d77c7d173aa","TopicArn":"arn:aws:s
east-1:333333333333:my-kinesis-test-topic","Subject":"My 3rd subject","Message":"My
 3rd body","Timestamp":"2020-11-27T00:31:39.755Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5"}
```


使用 Athena 分析存储在 Amazon S3 中的亚马逊SNS消息

本页介绍如何分析通过 Amazon Data Firehose 传送流SNS发送到亚马逊简单存储服务 (Amazon S3) 目标的亚马逊消息。

分析通过 Firehose SNS 发送到亚马逊 S3 目标的传输流的消息

1. 配置 Amazon S3 资源。有关说明，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶和 Amazon Simple Storage Service 用户指南中的使用 Amazon S3 存储桶。
2. 配置传输流。有关说明，请参阅亚马逊 D ata Firehose 开发者指南中的选择亚马逊 S3 作为您的目的地。
3. 使用 Amazon A thena 使用标准查询亚马逊 S3 对象。SQL有关更多信息，请参阅 Amazon Athena 用户指南中的入门。

示例查询

在本示例查询中，我们假设满足以下条件：

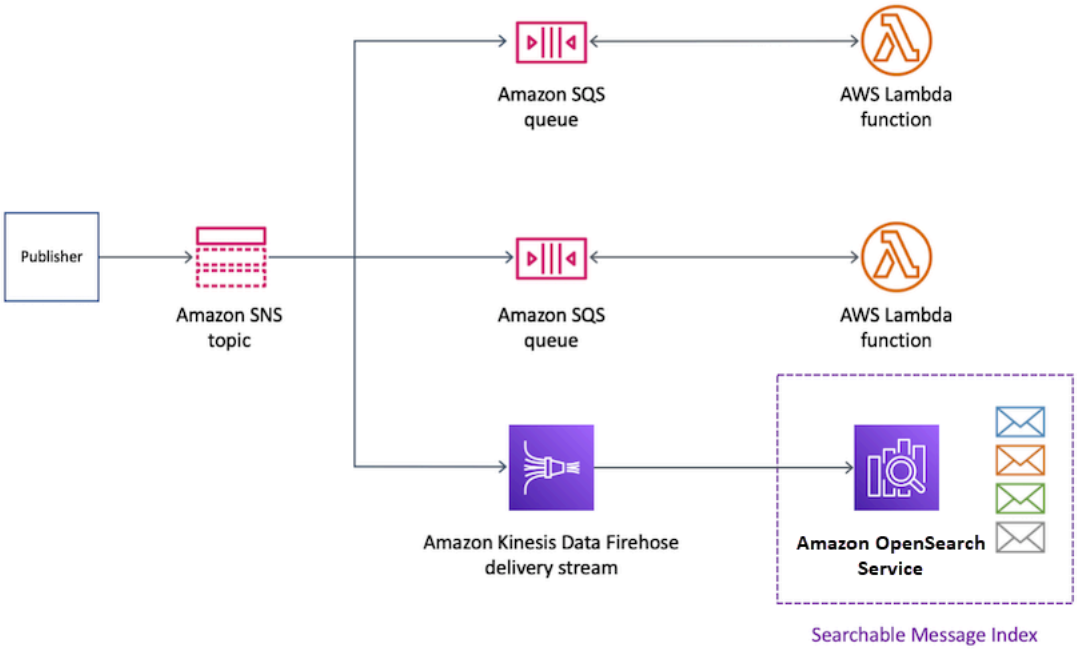
- 消息存储在 `default` schema 的 `notifications` 表中。
- `notifications` 表包含一个类型为 `string` 的 `timestamp` 列。

以下查询返回在指定日期范围内收到的所有SNS消息：

```
SELECT *
FROM default.notifications
WHERE from_iso8601_timestamp(timestamp) BETWEEN TIMESTAMP '2020-12-01 00:00:00' AND
 TIMESTAMP '2020-12-02 00:00:00';
```

将亚马逊SNS消息与亚马逊 OpenSearch 服务目标集成

本节提供有关向亚马逊 OpenSearch 服务（OpenSearch 服务）发布数据的亚马逊数据 Firehose 传输流的信息。



主题

- 在 OpenSearch 服务索引中存储和格式化 Amazon SNS 通知
- 分析 OpenSearch 服务目标的 Amazon SNS 消息

在 OpenSearch 服务索引中存储和格式化 Amazon SNS 通知

以下示例显示了向名为的亚马逊 OpenSearch 服务（OpenSearch 服务）索引发送的亚马逊SNS通知`my-index`。此索引在 `Timestamp` 字段中具有时间筛选字段。SNS通知位于有效载荷的`_source`属性中。

> **Note**
>
> 在此示例中，已发布消息的原始消息传输被禁用。禁用原始消息传送后，Amazon SNS 会向消息添加JSON元数据，包括以下属性：
>
> - Type
> - MessageId
> - TopicArn
> - Subject
> - Timestamp
> - UnsubscribeURL
> - MessageAttributes
>
> 有关原始消息传输的更多信息，请参阅 Amazon SNS 原始消息传送。


```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "4961310096311132320325040540219328379477388655098593280 2.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
      }
    }
  },
```

```
  "fields": {
    "Timestamp": [
      "2020-12-02T22:29:21.189Z"
    ]
  },
  "sort": [
    1606948161189
  ]
}
```

分析 OpenSearch 服务目标的 Amazon SNS 消息

本页介绍如何分析通过 Amazon Data Firehose 传送流发送到亚马逊 OpenSearch 服务（OpenSearch 服务）目的地的亚马逊SNS消息。

分析通过 Firehose SNS 发送到服务目标的传送流的 OpenSearch 消息

1. 配置您的 OpenSearch 服务资源。有关说明，请参阅《亚马逊 OpenSearch 服务开发者指南》中的 “亚马逊 OpenSearch 服务入门”。
2. 配置传输流。有关说明，请参阅 Amazon Data Firehose 开发者指南中的为您的目的地选择 OpenSearch 服务。
3. 使用 OpenSearch 服务查询和 Kibana 运行查询。有关更多信息，请参阅《亚马逊 OpenSearch 服务开发者指南》中的步骤 3：在 OpenSearch 服务域中搜索文档和 Kibana。

示例查询

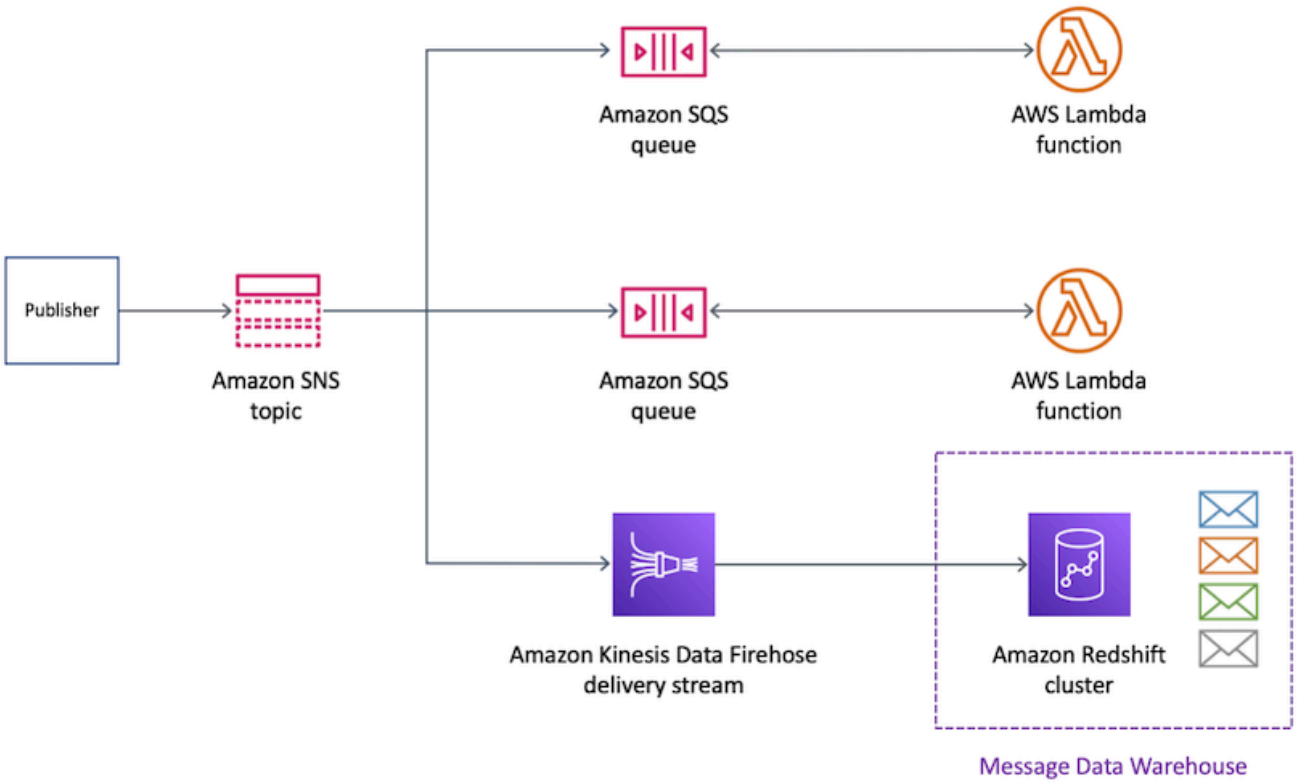
以下示例查询指定日期范围内收到的所有SNS消息的`my-index`索引：

```
POST https://search-my-domain.us-east-1.es.amazonaws.com/my-index/_search
{
  "query": {
    "bool": {
      "filter": [
        {
          "range": {
            "Timestamp": {
              "gte": "2020-12-08T00:00:00.000Z",
              "lte": "2020-12-09T00:00:00.000Z",
              "format": "strict_date_optional_time"
            }
          }
```

```
        }
      ]
    }
  }
}
```

在亚马逊 Redshift 目的地配置亚马逊SNS消息传输和分析

本节介绍如何将亚马逊SNS通知分散到向亚马逊 Redshift 发布数据的亚马逊 Data Firehose 传送流。通过此配置，您可以连接到 Amazon Redshift 数据库，并使用SQL查询工具在数据库中查询符合特定条件的亚马逊SNS消息。



主题

- 在 Amazon Redshi SNS ft 表中构造亚马逊邮件档案
- 分析存储在亚马逊 Redshift 目标中的亚马逊SNS消息

在 Amazon Redshi SNS ft 表中构造亚马逊邮件档案

对于 Amazon Redshift 终端节点，已发布的亚马逊SNS消息以行形式存档在表中。示例如下：

> **Note**
>
> 在此示例中，已发布消息的原始消息传输被禁用。禁用原始消息传送后，Amazon SNS 会向消息添加JSON元数据，包括以下属性：
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Message`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> 有关原始消息传输的更多信息，请参阅 Amazon SNS 原始消息传送。
> 尽管亚马逊使用此列表中显示的大写字母为消息SNS添加属性，但 Amazon Redshift 表中的列名全部以小写字符显示。要转换 Amazon Redshift 终端节点的JSON元数据，您可以使用命令。SQL `COPY`有关更多信息，请参阅 Amazon Redshift 数据库开发者指南中的从JSON示例复制和使用 “自动忽略大小写” 选项从JSON数据加载。

type	messageid	topicarn	subject	消息	时间戳	unsubscribeurl	messageattributes
Notification	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	示例主题	示例消息	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=unsubscribe&=arnSubscriptionAr	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	消息	时间戳	unsubscribeurl	messageattributes
						n: aws: sns: us-east-1 :111111111111:my-topic: 326deeeb-cbf4-45da-b92b-ca77a247813b	
Notification	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	示例主题 2	示例消息 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=unsubscribe&=arnSubscriptionArn: aws: sns: us-east-1 :111111111111:my-topic: 326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	消息	时间戳	unsubscribeurl	messageattributes
Notification	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	示例主题 3	示例消息 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=unsubscribe&=arnSubscriptionArn: aws:sns: us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

有关向 Amazon Redshift 终端节点扇出通知的更多信息，请参阅 在亚马逊 Redshift 目的地配置亚马逊SNS消息传输和分析。

分析存储在亚马逊 Redshift 目标中的亚马逊SNS消息

本页介绍如何分析通过亚马逊 Data Firehose 传送流SNS发送到亚马逊 Redshift 目的地的亚马逊消息。

分析通过 Firehose 传送流SNS发送到亚马逊 Redshift 目的地的消息

1. 配置您的 Amazon Redshift 资源。有关说明，请参阅 Amazon Redshift 入门指南中的Amazon Redshift 入门。
2. 配置传输流。有关说明，请参阅《亚马逊数据 Firehose 开发者指南》中的为目的地选择亚马逊 Redsh ift。
3. 运行查询。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 使用查询编辑器查询数据库。

示例查询

在本示例查询中，我们假设满足以下条件：

- 消息存储在默认 `public` schema 的 `notifications` 表中。
- SNS消息中的`Timestamp`属性存储在表的`timestamp`列中，列数据类型为`timestamptz`。

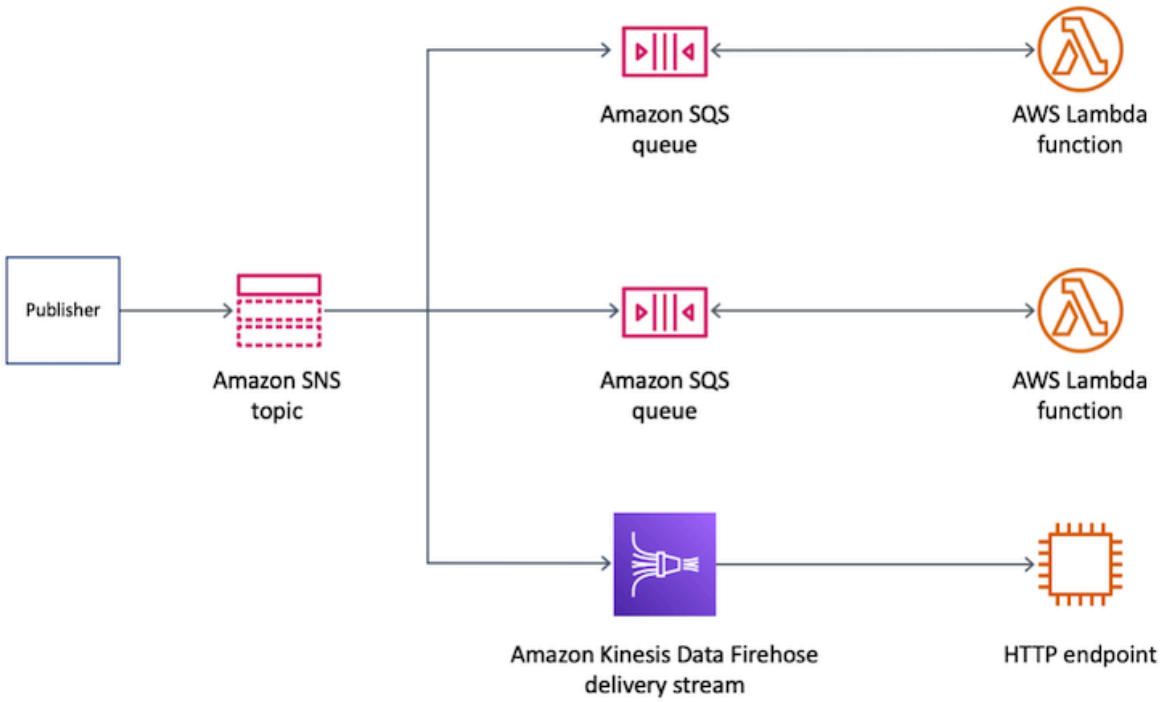
> **Note**
>
> 要转换 Amazon Redshift 终端节点的JSON元数据，您可以使用命令。SQL `COPY`有关更多信息，请参阅 Amazon Redshift 数据库开发者指南中的从JSON示例复制和使用 “自动忽略大小写” 选项从JSON数据加载。

以下查询返回在指定日期范围内收到的所有SNS消息：

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

使用 Amazon Data Firehose 配置亚马逊向HTTP目的地SNS发送消息

本节提供有关将数据发布到终端节点的 Amazon Data Firehose 传输流的信息。HTTP



主题

- 配送到HTTP目的地的 Amazon SNS 通知格式

配送到HTTP目的地的 Amazon SNS 通知格式

以下是来自亚马逊的HTTPPOST请求正文示例SNS，Amazon Data Firehose 传输流可以将其发送到终端节点。HTTP该SNS通知在属性中编码为 base64 有效负载。`records`

> **Note**
>
> 在此示例中，已发布消息的原始消息传输被禁用。有关原始消息传输的更多信息，请参阅 Amazon SNS 原始消息传送。

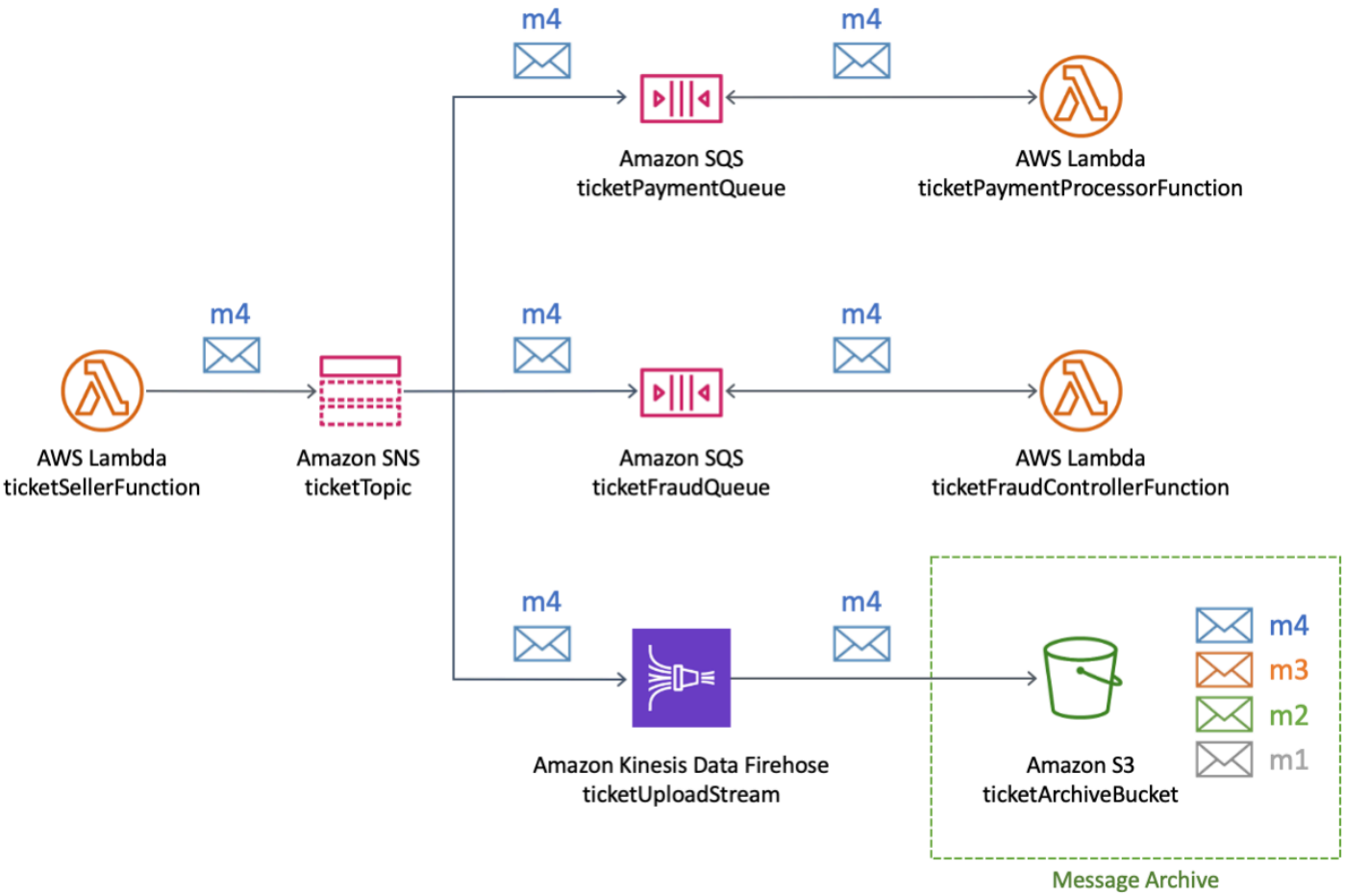
```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
    "records": [
      {
```

```
        "data":
"eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```


Amazon SNS 消息存档和分析：航空票务平台的示例用例

本节提供存档和分析 Amazon SNS 消息的常见用例教程。

此使用案例的设置是一个在受监管环境中运行的航空公司票务平台。该平台受合规性框架的约束，要求公司将所有售票记录存档至少五年。为了实现数据保留方面的合规目标，该公司订阅了现有主题的 Amazon Data Fireh SNS ose 传送流。传输流的目标是 Amazon Simple Storage Service (Amazon S3) 存储桶。使用此配置，发布到该SNS主题的所有事件都将存档在 Amazon S3 存储桶中。下图显示了此配置的架构：



为了进行分析并深入了解门票销售情况，该公司使用Amazon Athena进行SQL查询。例如，公司可以通过查询来了解最受欢迎的目的地和最频繁的旅客。

要为此用例创建 AWS 资源，您可以使用 AWS Management Console 或 AWS CloudFormation 模板。

主题

为 Amazon SNS 消息存档和分析设置初始 AWS 资源

本页面介绍如何为消息存档和分析示例使用案例创建以下资源：

- 一个 Amazon Simple Storage Service (Amazon S3) 存储桶
- 两个亚马逊简单队列服务 (AmazonSQS) 队列
- 亚马逊的SNS话题
- 两次亚马逊SQS订阅了亚马逊SNS话题

要创建起始资源

1. 创建 Amazon S3 存储桶：
   a. 打开 Amazon S3 控制台。
   b. 选择 Create bucket（创建存储桶）。
   c. 对于 Bucket name（存储桶名称），请输入全局唯一名称。保留其他字段作为默认值。
   d. 选择创建存储桶。

   有关 Amazon S3 存储桶的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶和 Amazon Simple Storage Service 用户指南中的使用 Amazon S3 存储桶。
2. 创建两个 Amazon SQS 队列：
   a. 打开 Amazon SQS 控制台。

b. 选择创建队列。

c. 对于 Type（类型），选择 Standard（标准）。

d. 对于名称，请输入 **ticketPaymentQueue**。

e. 在 Access policy（访问策略）下，对于 Choose method（选择方法），选择 Advanced（高级）。

f. 在JSON策略框中，粘贴以下策略：

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-1:123456789012:ticketTopic"
        }
      }
    }
  ]
}
```

在此访问策略中，替换 AWS 账户 数字 (*123456789012*) 用你自己的，然后更改 AWS 区域 (*us-east-1*) 相应地。

g. 选择创建队列。

h. 重复这些步骤以创建第二个名为的SQS队列**ticketFraudQueue**。

有关创建SQS队列的更多信息，请参阅《亚马逊简单SQS队列服务开发者指南》中的创建亚马逊队列（控制台）。

3. 创建SNS主题：

a. 打开 Amazon SNS 控制台的主题页面。

b. 选择创建主题。

c. 在 Details（详细信息）下，对于 Type（类型），选择 Standard（标准）。
d. 对于名称，请输入 **ticketTopic**。
e. 选择创建主题。

有关创建SNS主题的更多信息，请参阅创建亚马逊SNS主题。

4. 为两个SQS队列订阅该SNS主题：

a. 在 Amazon SNS 控制台的ticketTopic主题详情页面上，选择创建订阅。
b. 在 “详细信息” 下的 “协议” 中，选择 Amazon SQS。
c. 对于终端节点，选择ticketPaymentQueue队列的 Amazon 资源名称 (ARN)。
d. 选择创建订阅。
e. 重复这些步骤，使用ticketFraudQueue队列中的ARN创建第二个订阅。

有关订阅SNS主题的更多信息，请参阅创建对 Amazon SNS 主题的订阅。您也可以从 Amazon SQS 控制台为SNS主题SQS队列订阅。有关更多信息，请参阅《亚马逊简单SQS队列服务开发者指南》中的为亚马逊队列订阅亚马逊SNS主题（控制台）。

您已为此示例使用案例创建了初始资源。要继续，请参阅 为亚马逊消息存档设置 Firehose 传送流 SNS。

为亚马逊消息存档设置 Firehose 传送流 SNS

本页介绍如何为消息存档和分析示例用例创建 Amazon Data Firehose 传输流。

创建 Firehose 传送流

1. 打开 Amazon Kinesis 服务控制台。
2. 选择 Firehose，然后选择创建传送流。
3. 在 New delivery stream（新传输流）页面上，对于 Delivery stream name（传输流名称），输入 **ticketUploadStream**，然后选择 Next（下一步）。
4. 在 Process records（处理记录）页面上，选择 Next（下一步）。
5. 在 Choose a destination（选择目标）页面上，执行以下操作：

a. 对于 Destination（目标），选择 Amazon S3。
b. 在 S3 destination（S3 目标）下，对于 S3 bucket（S3 存储桶），选择您最初创建的 S3 存储桶。

c. 选择下一步。

6. 在 Configure settings（配置设置）页面上，对于 S3 缓冲区条件，执行以下操作：

   - 对于 Buffer size（缓冲区大小），输入 **1**。
   - 对于 Buffer interval（缓冲区时间间隔），输入 **60**。

   通过将这些值用于 Amazon S3 缓冲区，您可以快速测试配置。满足的第一个缓冲条件将触发至 S3 存储桶的数据传输。

7. 在配置设置页面上，对于权限，选择创建一个 AWS Identity and Access Management (IAM) 角色并自动分配所需权限。然后选择下一步。
8. 在 Review（审核）页面上，选择 Create delivery stream（创建传输流）。
9. 从 Kinesis Data Firehose 交付流页面中，选择您刚刚ticketUploadStream创建的传输流（ ）。在“详情” 选项卡上，记下直播的 Amazon 资源名称 (ARN) 以备后用。

有关创建传输流的更多信息，请参阅《亚马逊数据 Firehose 开发者指南》中的创建亚马逊数据 Firehose 传送流。有关创建IAM角色的更多信息，请参阅《IAM用户指南》中的创建角色以向 AWS 服务委派权限。

您已使用所需权限创建了 Firehose 传送流。要继续，请参阅 将 Firehose 直播订阅亚马逊话题 SNS。

将 Firehose 直播订阅亚马逊话题 SNS

本页面介绍如何为消息存档和分析示例使用案例创建以下资源：

- 允许亚马逊SNS订阅在 Amazon Data Firehose 传送流上记录的 AWS Identity and Access Management (IAM) 角色
- 该主题的 Firehose 直播订阅 SNS

为 Amazon SNS 订阅创建IAM角色

1. 打开IAM控制台的 “角色” 页面。
2. 选择 Create role（创建角色）。
3. 对于Select type of trusted entity（选择受信任实体的类型），选择 AWS service（服务）。
4. 在 “选择用例” 中，选择SNS。然后选择下一步：权限。
5. 选择下一步：标签。

6. 选择下一步：审核。
7. 在审核页面上，对于角色名称，输入 **ticketUploadStreamSubscriptionRole**。然后选择创建角色。
8. 创建角色后，选择其名称 (ticketUploadStreamSubscriptionRole)。
9. 在角色的 Summary（摘要）页面上，选择 Add inline policy（添加内联策略）。
10. 在创建策略页面上，选择JSON选项卡，然后将以下策略粘贴到框中：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:ListDeliveryStreams",
                "firehose:ListTagsForDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ticketUploadStream"
            ],
            "Effect": "Allow"
        }
    ]
}
```

在本政策中，替换 AWS 账户 数字 (*123456789012*) 用你自己的，然后更改 AWS 区域 (*us-east-1*) 相应地。

11. 选择查看策略。
12. 在 Create policy（创建策略）页面上，对于 Name（名称），输入 **FirehoseSnsPolicy**。然后选择创建策略。
13. 在角色的 “摘要” 页面上，记下角色ARN以备后用。

有关创建IAM角色的更多信息，请参阅《IAM用户指南》中的创建角色以向 AWS 服务委派权限。

在 Firehose 直播中订阅该主题 SNS

1. 打开 Amazon SNS 控制台的主题页面。
2. 在 Subscriptions（订阅）选项卡上，选择 Create subscription（创建订阅）。
3. 在 “详细信息” 下的 “协议” 中，选择 Amazon Data Firehose。
4. 在 En d point 中，输入您之前创建的ticketUploadStream传输流的 Amazon 资源名称 (ARN)。例如，输入 **`arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream`**。
5. 对于订阅角色 ARN，ARN请输入您之前创建的ticketUploadStreamSubscriptionRoleIAM 角色。例如，输入 **`arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole`**。
6. 选择 Enable raw message delivery（启用原始消息传输）复选框。
7. 选择创建订阅。

您已创建IAM角色和SNS主题订阅。要继续，请参阅 测试和查询 Amazon SNS 配置以实现有效的数据管理。

测试和查询 Amazon SNS 配置以实现有效的数据管理

本页介绍如何通过向 Amazon SNS 主题发布消息来测试消息存档和分析示例用例。这些说明包括一个示例查询，您可以运行并根据自己的需求对其进行调整。

测试配置

1. 打开 Amazon SNS 控制台的主题页面。
2. 选择 **`ticketTopic`** 主题。
3. 选择发布消息。
4. 在 Publish message to topic（发布消息到主题）页面上，为消息正文输入以下内容。在消息结尾处添加换行符。

   ```
   {"BookingDate":"2020-12-15","BookingTime":"2020-12-15
    04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
   ```

   保留所有其他选项作为默认值。
5. 选择发布消息。

有关发布消息的更多信息，请参阅 向 Amazon SNS 主题发布消息。

6. 在 60 秒的传输流间隔后，打开 Amazon Simple Storage Service (Amazon S3) 控制台并选择您最初创建的 Amazon S3 存储桶。

   存储桶中会显示已发布的消息。

要查询数据

1. 打开 Amazon Athena 控制台。
2. 运行查询。

   例如，假设 `default` schema 中的 `notifications` 表包含下列数据：

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 11:30:15","Destination":"Miami","FlyingFrom":"Omaha","TicketNumber":"efgh5678"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 3:30:10","Destination":"Miami","FlyingFrom":"NewYork","TicketNumber":"ijkl9012"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 12:30:05","Destination":"Delhi","FlyingFrom":"Omaha","TicketNumber":"mnop3456"}
```

   要查找最佳目标，请运行以下查询：

```
SELECT destination
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

   要查询在特定日期和时间范围内售出的票证，请运行如下所示的查询：

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
  AND TIMESTAMP '2020-12-15 12:00:00';
```

您可以根据自己的需求调整两个示例查询。有关使用 Athena 运行查询的更多信息，请参阅 Amazon Athena 用户指南中的入门。

清理

为避免在完成测试后产生使用费用，请删除您在本教程中创建的以下资源：

- 亚马逊SNS订阅
- 亚马逊SNS话题
- 亚马逊简单队列服务 (AmazonSQS) 队列
- Amazon S3 存储桶
- 亚马逊 Data Firehose 传送流
- AWS Identity and Access Management (IAM) 角色和政策

使用模板自动存档 Amazon SNS 消息 AWS CloudFormation

要自动部署 Amazon SNS 消息存档和分析示例用例，您可以使用以下YAML模板：

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Template for creating an SNS archiving use case
Resources:
  ticketUploadStream:
    DependsOn:
    - ticketUploadStreamRolePolicy
    Type: AWS::KinesisFirehose::DeliveryStream
    Properties:
      S3DestinationConfiguration:
        BucketARN: !Sub 'arn:${AWS::Partition}:s3:::${ticketArchiveBucket}'
        BufferingHints:
          IntervalInSeconds: 60
          SizeInMBs: 1
        CompressionFormat: UNCOMPRESSED
        RoleARN: !GetAtt ticketUploadStreamRole.Arn
  ticketArchiveBucket:
    Type: AWS::S3::Bucket
  ticketTopic:
    Type: AWS::SNS::Topic
  ticketPaymentQueue:
```

```
    Type: AWS::SQS::Queue
  ticketFraudQueue:
    Type: AWS::SQS::Queue
  ticketQueuePolicy:
    Type: AWS::SQS::QueuePolicy
    Properties:
      PolicyDocument:
        Statement:
          Effect: Allow
          Principal:
            Service: sns.amazonaws.com
          Action:
            - sqs:SendMessage
          Resource: '*'
          Condition:
            ArnEquals:
              aws:SourceArn: !Ref ticketTopic
      Queues:
        - !Ref ticketPaymentQueue
        - !Ref ticketFraudQueue
  ticketUploadStreamSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
      Endpoint: !GetAtt ticketUploadStream.Arn
      Protocol: firehose
      SubscriptionRoleArn: !GetAtt ticketUploadStreamSubscriptionRole.Arn
  ticketPaymentQueueSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
      Endpoint: !GetAtt ticketPaymentQueue.Arn
      Protocol: sqs
  ticketFraudQueueSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
      Endpoint: !GetAtt ticketFraudQueue.Arn
      Protocol: sqs
  ticketUploadStreamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
```

```
      Statement:
      - Sid: ''
        Effect: Allow
        Principal:
          Service: firehose.amazonaws.com
        Action: sts:AssumeRole
  ticketUploadStreamRolePolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: FirehoseticketUploadStreamRolePolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - s3:AbortMultipartUpload
          - s3:GetBucketLocation
          - s3:GetObject
          - s3:ListBucket
          - s3:ListBucketMultipartUploads
          - s3:PutObject
          Resource:
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}'
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}/*'
      Roles:
      - !Ref ticketUploadStreamRole
  ticketUploadStreamSubscriptionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - sns.amazonaws.com
          Action:
          - sts:AssumeRole
      Policies:
      - PolicyName: SNSKinesisFirehoseAccessPolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
          - Action:
```

```
            - firehose:DescribeDeliveryStream
            - firehose:ListDeliveryStreams
            - firehose:ListTagsForDeliveryStream
            - firehose:PutRecord
            - firehose:PutRecordBatch
            Effect: Allow
            Resource:
            - !GetAtt ticketUploadStream.Arn
```

Fanout Amazon SNS 向 Lambda 函数发送通知，用于自动处理

亚马逊SNS和 AWS Lambda 已集成，因此您可以使用亚马逊通知调用 Lambda 函数。SNS当消息发布到订阅了 Lambda 函数的SNS主题时，将使用已发布消息的有效负载调用 Lambda 函数。Lambda 函数将消息负载作为输入参数接收，并且可以操作消息中的信息、将消息发布到其他SNS主题或将消息发送到其他 AWS 服务。

Amazon SNS 还支持发送到 Lambda 终端节点的消息通知的消息传输状态属性。有关更多信息，请参阅 Amazon SNS 消息传送状态。

主题

- 跨区域将 Amazon SNS 与 Lambda 函数集成的先决条件
- 将 Lambda 函数订阅到亚马逊主题 SNS

跨区域将 Amazon SNS 与 Lambda 函数集成的先决条件

要使用亚马逊SNS通知调用 Lambda 函数，您需要满足以下条件：

- Lambda 函数
- 亚马逊SNS话题

有关创建用于亚马逊的 Lambda 函数的信息SNS，请参阅在亚马逊上使用 Lambda。SNS有关创建 Amazon SNS 主题的信息，请参阅创建主题。

当您使用 Amazon SNS 将来自选择加入区域的消息传送到默认启用的区域时，您必须将委托人替换为，从而更改在 Lamb AWS da 函数中创建的`sns.amazonaws.com`策略。`sns.<opt-in-region>.amazonaws.com`

例如，如果您想将美国东部（弗吉尼亚北部）的 Lambda 函数订阅到亚太地区（香港）的某个SNS主题，请将 Lambda AWS 函数策略中的委托人更改为。sns.ap-east-1.amazonaws.com选择加入的区域包括 2019 年 3 月 20 日之后推出的任何区域，包括亚太地区（香港）、中东（巴林）、欧盟（米兰）和非洲（开普敦）。2019 年 3 月 20 日之前推出的区域默认情况下处于启用状态。

> **Note**
>
> AWS 不支持从默认启用的区域向 Lambda 跨区域交付到可选区域。此外，不支持跨区域将SNS邮件从选择加入区域转发到其他选择加入区域。

将 Lambda 函数订阅到亚马逊主题 SNS

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 在主题 页上，选择一个主题。
4. 在订阅部分中，选择创建订阅。
5. 在 Create subscription（创建订阅）页上的 Details（详细信息）部分中，执行以下操作：
   a. 验证所选主题ARN。
   b. 对于协议，请选择 AWS Lambda。
   c. 对于 En d point，输入函数的。ARN
   d. 选择创建订阅。

当消息发布到订阅了 Lambda 函数的SNS主题时，将使用已发布消息的有效负载调用 Lambda 函数。有关如何在亚马逊SNS上使用的 AWS Lambda 信息（包括教程），请参阅在亚马逊 AWS Lambda 上使用SNS。

Fanout Amazon SNS 通知到亚马逊SQS队列进行异步处理

亚马逊SNS与亚马逊简单队列服务（亚马逊SQS）密切合作。这些服务为开发人员提供了不同的益处。Amazon SNS 允许应用程序通过 “推送” 机制向多个订阅者发送时间紧迫的消息，无需定期检查或 “轮询” 更新。Amazon SQS 是一项消息队列服务，分布式应用程序使用它通过轮询模型交换消息，并且可用于分离发送和接收组件，而无需每个组件同时可用。SQS结合使用 Amazon SNS 和 Amazon，可以将消息传送到需要立即通知事件的应用程序，也可以保留在亚马逊SQS队列中，供其他应用程序稍后处理。

当您为亚马逊SQS队列订阅亚马逊SNS主题时，您可以向该主题发布消息，然后亚马逊SNS会向已订阅的队列发送一条亚马逊SQS消息。Amazon SQS 消息包含发布到该主题的主题和消息，以及JSON文档中有关该消息的元数据。Amazon SQS 消息将与以下JSON文档类似。

```
{
   "Type" : "Notification",
   "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
   "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
   "Subject" : "Testing publish to subscribed queues",
   "Message" : "Hello world!",
   "Timestamp" : "2012-03-29T05:12:16.901Z",
   "SignatureVersion" : "1",
   "Signature" : "EXAMPLEnTrFPa3...",
   "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:MyTopic:c7fe3a54-
ab0e-4ec2-88e0-db410a0f2bee"
}
```

为亚马逊SQS队列订阅亚马逊主题 SNS

要允许亚马逊SNS主题向亚马逊SQS队列发送消息，请执行以下任一操作：

- 使用 Amazon SQS 控制台，这样可以简化流程。有关更多信息，请参阅《亚马逊简单SQS队列服务开发者指南》中的 “为亚马逊队列订阅亚马逊队列” SNS 主题。
- 按照以下步骤进行操作：
  1. 获取您要向其发送消息的队列的 Amazon 资源名称 (ARN) 以及您要订阅该队列的主题。
  2. 向 Amazon SNS 主题`sqs:SendMessage`授予权限，使其可以向队列发送消息。
  3. 在队列中订阅 Amazon SNS 主题。
  4. 授予IAM用户或 AWS 账户 相应的权限，使其能够发布到亚马逊SNS主题并阅读来自亚马逊SQS队列的消息。
  5. 通过向主题发布消息，并读取来自队列的消息，对其进行测试。

要了解如何设置主题以向位于不同的 AWS账户中的队列发送消息，请参阅使用其他账户向亚马逊SQS队列发送亚马逊SNS消息。

要查看用于创建向两个队列发送消息的主题的 AWS CloudFormation 模板，请参阅通过以下方式自动SQS发送亚马逊SNS到亚马逊的消息 AWS CloudFormation。

第 1 步：获取队ARN列和主题

在为队列订阅你的主题时，你需要一份该队列ARN的副本。同样，在授予主题向队列发送消息的权限时，您需要该ARN主题的副本。

要获取队列ARN，您可以使用 Amazon SQS 控制台或GetQueueAttributesAPI操作。

ARN从 Amazon SQS 控制台获取队列

1. 登录 AWS Management Console 并打开 Amazon SQS 控制台，网址为https://console.aws.amazon.com/sqs/。
2. 选中要获取的队列的ARN复选框。
3. 从 “详情” 部分复制该ARN值，以便您可以使用它来订阅 Amazon SNS 主题。

要获取主题ARN，您可以使用 Amazon SNS 控制台、`sns-get-topic-attributes`命令或`GetQueueAttributes`API操作。

要ARN从 Amazon SNS 控制台获取主题

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择ARN要获取的主题。
3. 从 “详情” 部分复制该ARN值，以便您可以使用它来授予 Amazon SNS 主题向队列发送消息的权限。

第 2 步：向亚马逊SNS主题授予向亚马逊SQS队列发送消息的权限

为了使亚马逊SNS主题能够向队列发送消息，您必须在队列上设置允许该亚马逊SNS主题执行`sqs:SendMessage`操作的策略。

获取主题和队列之后，方可为队列订阅主题。如果您尚未创建主题或队列，请您立刻创建。有关更多信息，请参阅创建主题，并参阅《Amazon Simple Queue Service 开发人员指南》中的创建队列。

要对队列设置策略，您可以使用 Amazon SQS 控制台或SetQueueAttributesAPI操作。在开始之前，请确保您拥有要允许向队列发送消息的主题的。ARN如果要为队列订阅多个主题，策略必须对于每个主题包含一个 `Statement` 元素。

使用 Amazon SQS 控制台对队列设置 SendMessage 策略

1. 登录 AWS Management Console 并打开 Amazon SQS 控制台，网址为https://console.aws.amazon.com/sqs/。
2. 选择要设置其策略的队列的框，然后依次选择 Access policy（访问策略）选项卡、编辑。
3. 在 Access policy（访问策略）部分中，定义谁可以访问您的队列。
   - 添加允许主题操作的一项条件。
   - 设置`Principal`为 Amazon SNS 服务，如下例所示。
   - 使用 `aws:SourceArn` 或 `aws:SourceAccount` 全局条件键来防止混淆代理场景。要使用这些条件键，请将值设置ARN为主题的。如果您的队列订阅了多个主题，则可以改用`aws:SourceAccount`。

   例如，以下策略允许 MyTopic 向发送消息 MyQueue。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
        }
      }
    }
  ]
}
```

第 3 步：为队列订阅 Amazon SNS 主题

要通过主题向队列发送消息，您必须为队列订阅 Amazon SNS 主题。您可以通过队列来指定队列ARN。要订阅主题，您可以使用 Amazon SNS 控制台、`sns-subscribe`CLI命令或`Subscribe`API操作。在开始之前，请确保你有要订阅的队列。ARN

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 在主题 页上，选择一个主题。
4. 在 ***MyTopic*** 页面上，在 “订阅” 页面中，选择 “创建订阅”。
5. 在 Create subscription（创建订阅）页上的 Details（详细信息）部分中，执行以下操作：

   a. 验证主题ARN。
   b. 对于协议，请选择 Amazon SQS。
   c. 对于终端节点，输入 Amazon SQS 队列的。ARN
   d. 选择创建订阅。

   确认订阅后，您新建订阅的“Subscription ID”将显示其订阅 ID。如果订阅由队列所有者创建，则订阅将自动确认，且订阅立刻可用。

   一般情况下，您可以在您自己的账户中为您的队列订阅您自己的主题。但是，您还可以通过另一账户为队列订阅主题。如果创建订阅的用户并非队列所有者（例如，如果账户 A 的用户为账户 B 中的队列订阅账户 A 中的主题），则必须对订阅进行确认。有关通过不同账户订阅队列和确认订阅的更多信息，请参阅 使用其他账户向亚马逊SQS队列发送亚马逊SNS消息。

步骤 4：向用户授予对适当主题和队列操作的权限

您应使用 AWS Identity and Access Management (IAM) 仅允许相应的用户向亚马逊SNS主题发布消息以及从亚马逊SQS队列中读取/删除消息。有关控制IAM用户主题和队列操作的更多信息在 Amazon 上使用基于身份的政策 SNS，请参阅《亚马逊简单队列服务开发者指南》SQS中的 “亚马逊中的身份和访问管理”。

可以采取两种方式控制对主题或队列的访问：

- 向IAM用户或组添加策略。为用户授予主题或队列权限的最简单方式就是创建群组，并为该群组添加适当策略，然后向此群组添加用户。相比较而言，向群组添加或删除用户，比追踪您为单独用户而设定的各项策略要简单得多。
- 添加策略至主题或队列。如果您想将某个主题或队列的权限授予其他 AWS 账户，那么唯一的方法就是添加一个以 AWS 账户 您想要授予权限的委托人为其委托人的策略。

绝大多数情况下，您应使用第一种方法（通过向群组添加或删除适当用户的方式，向群组添加策略，管理用户权限）。如果您需要向另一账户的用户授予权限，那么应使用第二种方法。

向IAM用户或组添加策略

如果您向IAM用户或组添加了以下策略，则需要向该用户或群组中的成员授予对该主题执行`sns:Publish`操作的权限 MyTopic。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

如果您向IAM用户或组添加了以下策略，则需要向该用户或组中的成员授予对队列 MyQueue 1 `sqs:ReceiveMessage` 和 MyQueue 2 执行和`sqs:DeleteMessage`操作的权限。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue1",
        "arn:aws:sqs:us-east-2:123456789012:MyQueue2"
      ]
    }
  ]
}
```

添加策略至主题或队列

以下策略示例显示如何为主题和队列授予另一账户授权。

> **Note**
>
> 当您向其他用户授予对您账户中某项资源的 AWS 账户 访问权限时，也就是向拥有该资源的管理员级访问权限（通配符访问权限）的IAM用户授予权限。其他账户中的所有其他IAM用户都

会被自动拒绝访问您的资源。如果您想向该IAM用户授予 AWS 账户 访问您资源的权限，则该账户或具有管理员级别访问权限的IAM用户必须将资源权限委托给这些IAM用户。有关跨账户委托的更多信息，请参阅《使用IAM指南》中的启用跨账户访问权限。

如果您向账户 123456789012 MyTopic 中的某个主题添加了以下策略，则需要向账户 111122223333 授予对该主题执行操作的权限。`sns:Publish`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

如果您向账户 123456789012 MyQueue 中的队列添加了以下策略，则需要授予账户 111122223333 对该队列执行和操作的权限。`sqs:ReceiveMessage` `sqs:DeleteMessage`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

步骤 5：测试主题的队列订阅

通过发布主题，查看主题向队列发送的消息，可以测试主题的队列订阅情况。

使用 Amazon SNS 控制台向主题发布内容

1. 使用 AWS 账户 或有权发布主题的IAM用户的证书，登录 AWS Management Console 并打开亚马逊SNS控制台，网址为https://console.aws.amazon.com/sns/。
2. 在导航面板上，选择主题，然后选择发布到主题。
3. 在主题框中，输入主题（例如 **`Testing publish to queue`**），在消息框中，输入一些文字（例如 **`Hello world!`**），然后选择发布消息。界面将显示如下消息：“Your message has been successfully published”（您的消息已成功发布）。

使用 Amazon SQS 控制台查看来自该主题的消息

1. 使用 AWS 账户 或有权查看队列中消息的IAM用户的证书，登录 AWS Management Console 并打开亚马逊SQS控制台，网址为https://console.aws.amazon.com/sqs/。
2. 选择已订阅该主题的 queue（队列）。
3. 选择 Send and receive messages（发送和接收消息），然后选择 Poll for messages（轮询消息）。界面将显示“Notification”类型消息。
4. 在正文列中，选择更多详细信息。“消息详细信息” 框包含一个JSON文档，其中包含您向该主题发布的主题和消息。该消息与以下JSON文档类似。

```
{
  "Type" : "Notification",
  "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "Testing publish to subscribed queues",
  "Message" : "Hello world!",
  "Timestamp" : "2012-03-29T05:12:16.901Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEnTrFPa3...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c7fe3a54-ab0e-4ec2-88e0-db410a0f2bee"
}
```

5. 选择关闭。您已经成功发布到一个主题，该主题向队列发送通知消息。

通过以下方式自动SQS发送亚马逊SNS到亚马逊的消息 AWS CloudFormation

AWS CloudFormation 允许您使用模板文件将 AWS 资源集合作为一个单元一起创建和配置。通过本部分提供的模板示例，您可以轻松部署向队列发布的主题。这些模板通过创建两个队列、创建订阅队列的主题、向队列添加策略以便主题可以向队列发送消息，以及创建IAM用户和组来控制对这些资源的访问权限，从而为您完成设置步骤。

有关使用 AWS CloudFormation 模板部署 AWS 资源的更多信息，请参阅《AWS CloudFormation 用户指南》中的 “入门”。

使用 AWS CloudFormation 模板在中设置主题和队列 AWS 账户

示例模板创建了一个 Amazon SNS 主题，该主题可以向两个 Amazon SQS 队列发送消息，并允许一个IAM群组的成员向该主题发布消息，另一个群组的成员可以从队列中读取消息。该模板还会创建添加到每个群组的IAM用户。

您可以将模板内容复制到文件中，您也可以从 “模板” 页面下载AWS CloudFormation 模板。在模板页面上，选择按 AWS 服务浏览示例模板，然后选择亚马逊简单队列服务。

M 设置ySNSTopic 为发布到两个已订阅的终端节点，即两个 Amazon SQS 队列（MyQueue1 和 MyQueue 2）。 MyPublishTopicGroup 是一个IAM群组，其成员有权ySNSTopic 使用 “发布” API 操作或 sns-p ub lish 命令向 M 发布内容。该模板创建IAM用户， MyPublishUser MyQueueUser 并为他们提供登录配置文件和访问密钥。通过该模板创建堆栈的用户将指定登录界面密码作为输入参数。该模板使用 MyPublishUserKey 和为两个IAM用户创建访问密钥 MyQueueUserKey。 AddUserToMyPublishTopicGroup 添加 MyPublishUser 到， MyPublishTopicGroup 这样用户就可以拥有分配给该组的权限。

M yRDMessage QueueGroup 是一个IAM群组，其成员有权使用和DeleteMessageAPI操作从两个 Amazon SQS 队列中读取ReceiveMessage和删除消息。 AddUserToMyQueueGroup 添加 MyQueueUser 到 M 中，yRDMessageQueueGroup 这样用户就可以拥有分配给该组的权限。 MyQueuePolicy 为 M 分配ySNSTopic 向这两个队列发布通知的权限。

以下清单显示了 AWS CloudFormation 模板内容。

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
```

```
 "Description" : "AWS CloudFormation Sample Template SNSToSQS: This Template creates
an SNS topic that can send messages to
 two SQS queues with appropriate permissions for one IAM user to publish to the topic
and another to read messages from the queues.
 MySNSTopic is set up to publish to two subscribed endpoints, which are two SQS queues
(MyQueue1 and MyQueue2). MyPublishUser is an IAM user
 that can publish to MySNSTopic using the Publish API. MyTopicPolicy assigns that
permission to MyPublishUser. MyQueueUser is an IAM user
 that can read messages from the two SQS queues. MyQueuePolicy assigns those
permissions to MyQueueUser. It also assigns permission for
 MySNSTopic to publish its notifications to the two queues. The template creates
access keys for the two IAM users with MyPublishUserKey
 and MyQueueUserKey. ***Warning*** you will be billed for the AWS resources used if
you create a stack from this template.",

 "Parameters": {
   "MyPublishUserPassword": {
     "NoEcho": "true",
     "Type": "String",
     "Description": "Password for the IAM user MyPublishUser",
     "MinLength": "1",
     "MaxLength": "41",
     "AllowedPattern": "[a-zA-Z0-9]*",
     "ConstraintDescription": "must contain only alphanumeric characters."
   },
   "MyQueueUserPassword": {
     "NoEcho": "true",
     "Type": "String",
     "Description": "Password for the IAM user MyQueueUser",
     "MinLength": "1",
     "MaxLength": "41",
     "AllowedPattern": "[a-zA-Z0-9]*",
     "ConstraintDescription": "must contain only alphanumeric characters."
   }
 },


 "Resources": {
   "MySNSTopic": {
     "Type": "AWS::SNS::Topic",
     "Properties": {
       "Subscription": [{
           "Endpoint": {
             "Fn::GetAtt": ["MyQueue1", "Arn"]
```

```
          },
          "Protocol": "sqs"
        },
        {
          "Endpoint": {
            "Fn::GetAtt": ["MyQueue2", "Arn"]
          },
          "Protocol": "sqs"
        }
      ]
    }
  },
  "MyQueue1": {
    "Type": "AWS::SQS::Queue"
  },
  "MyQueue2": {
    "Type": "AWS::SQS::Queue"
  },
  "MyPublishUser": {
    "Type": "AWS::IAM::User",
    "Properties": {
      "LoginProfile": {
        "Password": {
          "Ref": "MyPublishUserPassword"
        }
      }
    }
  },
  "MyPublishUserKey": {
    "Type": "AWS::IAM::AccessKey",
    "Properties": {
      "UserName": {
        "Ref": "MyPublishUser"
      }
    }
  },
  "MyPublishTopicGroup": {
    "Type": "AWS::IAM::Group",
    "Properties": {
      "Policies": [{
        "PolicyName": "MyTopicGroupPolicy",
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
```

```
            "Action": [
              "sns:Publish"
            ],
            "Resource": {
              "Ref": "MySNSTopic"
            }
          }]
        }
      }]
    }
  },
  "AddUserToMyPublishTopicGroup": {
    "Type": "AWS::IAM::UserToGroupAddition",
    "Properties": {
      "GroupName": {
        "Ref": "MyPublishTopicGroup"
      },
      "Users": [{
        "Ref": "MyPublishUser"
      }]
    }
  },
  "MyQueueUser": {
    "Type": "AWS::IAM::User",
    "Properties": {
      "LoginProfile": {
        "Password": {
          "Ref": "MyQueueUserPassword"
        }
      }
    }
  },
  "MyQueueUserKey": {
    "Type": "AWS::IAM::AccessKey",
    "Properties": {
      "UserName": {
        "Ref": "MyQueueUser"
      }
    }
  },
  "MyRDMessageQueueGroup": {
    "Type": "AWS::IAM::Group",
    "Properties": {
      "Policies": [{
```

```
        "PolicyName": "MyQueueGroupPolicy",
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Action": [
              "sqs:DeleteMessage",
              "sqs:ReceiveMessage"
            ],
            "Resource": [{
                "Fn::GetAtt": ["MyQueue1", "Arn"]
              },
              {
                "Fn::GetAtt": ["MyQueue2", "Arn"]
              }
            ]
          }]
        }
      }]
    }
  },
  "AddUserToMyQueueGroup": {
    "Type": "AWS::IAM::UserToGroupAddition",
    "Properties": {
      "GroupName": {
        "Ref": "MyRDMessageQueueGroup"
      },
      "Users": [{
        "Ref": "MyQueueUser"
      }]
    }
  },
  "MyQueuePolicy": {
    "Type": "AWS::SQS::QueuePolicy",
    "Properties": {
      "PolicyDocument": {
        "Statement": [{
          "Effect": "Allow",
          "Principal": {
            "Service": "sns.amazonaws.com"
          },
          "Action": ["sqs:SendMessage"],
          "Resource": "*",
          "Condition": {
            "ArnEquals": {
```

```
              "aws:SourceArn": {
                "Ref": "MySNSTopic"
              }
            }
          }
        }]
      },
      "Queues": [{
        "Ref": "MyQueue1"
      }, {
        "Ref": "MyQueue2"
      }]
    }
  }
},
"Outputs": {
  "MySNSTopicTopicARN": {
    "Value": {
      "Ref": "MySNSTopic"
    }
  },
  "MyQueue1Info": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
          {
            "Fn::GetAtt": ["MyQueue1", "Arn"]
          },
          "URL:",
          {
            "Ref": "MyQueue1"
          }
        ]
      ]
    }
  },
  "MyQueue2Info": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
```

```
          {
            "Fn::GetAtt": ["MyQueue2", "Arn"]
          },
          "URL:",
          {
            "Ref": "MyQueue2"
          }
        ]
      ]
    }
  },
  "MyPublishUserInfo": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
          {
            "Fn::GetAtt": ["MyPublishUser", "Arn"]
          },
          "Access Key:",
          {
            "Ref": "MyPublishUserKey"
          },
          "Secret Key:",
          {
            "Fn::GetAtt": ["MyPublishUserKey", "SecretAccessKey"]
          }
        ]
      ]
    }
  },
  "MyQueueUserInfo": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
          {
            "Fn::GetAtt": ["MyQueueUser", "Arn"]
          },
          "Access Key:",
          {
            "Ref": "MyQueueUserKey"
```

```
            },
            "Secret Key:",
            {
              "Fn::GetAtt": ["MyQueueUserKey", "SecretAccessKey"]
            }
          ]
        ]
      }
    }
  }
}
```

Fanout Amazon SNS 向 HTTP /S 端点发送通知

您可以使用 Amazon SNS 向一个HTTP或多个HTTPS终端节点发送通知消息。当您为终端节点订阅主题时，您可以向该主题发布通知，然后 Amazon SNS 会发送HTTPPOST请求，将通知内容传送到已订阅的终端节点。当您订阅终端节点时，您可以选择 Amazon 是SNS使用HTTP还是HTTPS向终端节点发送POST请求。如果您使用HTTPS，则可以利用 Amazon 提供的以下SNS支持：

- 服务器名称指示 (SNI)-这允许 Amazon 支持SNS需要的HTTPS终端节点SNI，例如需要多个证书才能托管多个域的服务器。有关的更多信息SNI，请参阅服务器名称指示。

- 基本和摘要访问身份验证-这允许您在中HTTPSURL为HTTPPOST请求指定用户名和密码，例如`https://user:password@domain.com`或`https://user@domain.com`。用户名和密码通过使用HTTPS时建立的SSL连接进行加密。只有域名以明文形式发送。有关基本和摘要访问身份验证的更多信息，请参阅 RFC-2617。

> **Important**
>
> Amazon 目前SNS不支持私有 HTTP (S) 终端节点。
> HTTPSURLs只有您已授予访问权限的委托人才能从 Amazon SNS `GetSubscriptionAttributes` API 操作中检索。API

> **Note**
>
> 客户端服务必须能够支持 `HTTP/1.1 401 Unauthorized` 标头响应

该请求包含发布到该主题的主题和消息，以及JSON文档中有关通知的元数据。该请求将与以下HTTPPOST请求类似。有关HTTP标头和请求正文JSON格式的详细信息，请参阅HTTP/HTTPS标题和HTTP/HTTPS通知JSON格式。

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: da41e39f-ea4d-435a-b922-c6aae3915ebe
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
    Content-Length: 761
    Content-Type: text/plain; charset=UTF-8
    Host: ec2-50-17-44-49.compute-1.amazonaws.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "da41e39f-ea4d-435a-b922-c6aae3915ebe",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "test",
  "Message" : "test message",
  "Timestamp" : "2012-04-25T21:49:25.719Z",
  "SignatureVersion" : "1",
  "Signature" :
 "EXAMPLElDMXvB8r9R83tGoNn0ecwd5UjllzsvSvbItzfaMpN2nk5HVSw7XnOn/49IkxDKz8YrlH2qJXj2iZB0Zo2O71c4
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55"
}
```

主题

- 为 HTTP /S 终端节点订阅 Amazon 主题 SNS
- 验证 Amazon SNS 消息的签名
- 解析 Amazon SNS 消息格式

为 HTTP /S 终端节点订阅 Amazon 主题 SNS

本节中的页面介绍如何为 HTTP /S 终端节点订阅 Amazon SNS 主题。

主题

第 1 步：确保您的终端节点已准备就绪，可以处理 Amazon SNS 消息

在您HTTP或HTTPS终端节点订阅主题之前，必须确保HTTP或HTTPS终端节点能够处理 Amazon SNS 用于发送订阅确认和通知消息的HTTPPOST请求。通常，这意味着要创建和部署一个处理HTTP来自亚马逊的请求的网络应用程序（例如，如果您的终端节点主机运行的是带有 Apache 和 Tomcat 的 Linux，则为 Java servlet）。SNS当您订阅HTTP终端节点时，Amazon SNS 会向其发送订阅确认请求。创建订阅时，您的终端节点必须准备好接收和处理此请求，因为亚马逊当时SNS会发送此请求。在您确认订阅之前，Amazon SNS 不会向终端节点发送通知。确认订阅后，当对订阅的主题执行发布操作时，Amazon SNS 将向终端节点发送通知。

设置您的终端节点，处理订阅确认和通知消息

1. 您的代码应读取 Amazon SNS 发送到您的终端节点的HTTPPOST请求HTTP标头。您的代码应查找标题字段`x-amz-sns-message-type`，该字段会告诉您 Amazon SNS 发送给您的消息类型。通过查看标头，您无需解析HTTP请求正文即可确定消息类型。您需要处理如下两种类型消息：`SubscriptionConfirmation` 和 `Notification`。仅当从主题中删除订阅时，方使用 `UnsubscribeConfirmation` 消息。

   有关HTTP标题的详细信息，请参阅HTTP/HTTPS标题。以下HTTPPOST请求是订阅确认消息的示例。

```
POST / HTTP/1.1
    x-amz-sns-message-type: SubscriptionConfirmation
    x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
```

```
    Content-Length: 1336
    Content-Type: text/plain; charset=UTF-8
    Host: example.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37f...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH+...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

2. 您的代码应解析HTTPPOST请求正JSON文中的文档，然后使用内容键入 text/plain 来读取构成 Amazon 消息的名称/值对。SNS使用JSON解析器将控制字符的转义表示形式转换回其ASCII字符值（例如，将\ n 转换为换行符）。你可以使用现有的JSON解析器，比如 Jacks on JSON 处理器，也可以自己编写。为了使主题和消息字段中的文本有效JSON，Amazon SNS 必须将某些控制字符转换为可以包含在JSON文档中的转义表示形式。当您收到发送到终端节点的POST请求正文中的JSON文档时，如果您想精确表示原始主题和发布到主题的消息，则必须将转义字符转换回其原始字符值。由于签名采用了原始形式的消息和主题作为待签字符串的一部分，因此如果您想要验证通知签名，则上述操作非常重要。

3. 您的代码应验证 Amazon SNS 发送的通知、订阅确认或取消订阅确认消息的真实性。使用亚马逊SNS消息中包含的信息，您的终端节点可以重新创建签名，这样您就可以通过将您的签名与亚马逊随消息一起SNS发送的签名进行匹配来验证消息的内容。有关验证消息签名的更多信息，请参阅 验证 Amazon SNS 消息的签名。

4. 根据标题字段指定的类型`x-amz-sns-message-type`，您的代码应读取HTTP请求正文中包含的JSON文档并处理消息。这里是处理两大主要消息类型的指导原则：

SubscriptionConfirmation

阅读的值SubscribeURL并访问该值URL。要确认订阅并开始在终端节点上接收通知，您必须访问 SubscribeURLURL（例如，向发送HTTPGET请求URL）。请查看上一步中的示例HTTP请求，了解其SubscribeURL外观。有关 SubscriptionConfirmation 消息格式的更多信息，请参阅 HTTP/HTTPS订阅确认JSON格式。当你访问时URL，你会得到一个类似于以下XML文档的回复。该文档返回ConfirmSubscriptionResult元素内端点ARN的订阅。

```
<ConfirmSubscriptionResponse xmlns="http://sns.amazonaws.com/doc/2010-03-31/">
   <ConfirmSubscriptionResult>
      <SubscriptionArn>arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55</
SubscriptionArn>
   </ConfirmSubscriptionResult>
   <ResponseMetadata>
      <RequestId>075ecce8-8dac-11e1-bf80-f781d96e9307</RequestId>
   </ResponseMetadata>
</ConfirmSubscriptionResponse>
```

除了访问之外SubscribeURL，您还可以使用在SubscriptionConfirmation消息中Token设置相应值的ConfirmSubscription操作来确认订阅。如果您仅允许主题所有者和订阅所有者拥有取消订阅终端节点的权限，那么您可以通过 AWS 签名调用 ConfirmSubscription 操作。

通知

读取 Subject 和 Message 值，获取已向主题发布的通知信息。

有关 Notification 消息格式的详细信息，请参阅 HTTP/HTTPS标题。以下HTTPPOST请求是发送到终端节点 example.com 的通知消息的示例。

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
    Content-Length: 773
    Content-Type: text/plain; charset=UTF-8
    Host: example.com
    Connection: Keep-Alive
```

```
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
}
```

5. 确保您的终端节点使用相应的状态代码响应来自亚马逊SNS的HTTPPOST消息。此项连接将在 15 秒内超时。如果您的终端节点在连接超时之前没有响应，或者您的终端节点返回的状态代码超出了 200—4 xx 的范围，Amazon SNS 会将消息的传递视为失败的尝试。
6. 请确保您的代码可以处理 Amazon SNS 的消息传送重试。如果 Amazon SNS 未收到来自您的终端节点的成功响应，它将尝试再次传送消息。这适用于包括订阅确认消息在内的所有消息。默认情况下，如果消息的初始传送失败，Amazon 最多会SNS尝试三次重试，失败尝试之间的延迟设置为 20 秒。

   > **Note**
   >
   > 消息请求 15 秒后超时。这意味着，如果消息传送失败是由超时造成的，Amazon 将在上SNS次尝试传送后重试大约 35 秒。您可以为终端节点设置不同的发送策略。

   亚马逊SNS使用标`x-amz-sns-message-id`题字段来唯一标识发布到亚马逊SNS主题的每条消息。通过将已处理IDs的消息与传入的消息进行比较，可以确定该消息是否为重试尝试。
7. 如果您要订阅HTTPS终端节点，请确保您的终端节点具有来自可信证书颁发机构 (CA) 的服务器证书。亚马逊只SNS会向拥有由亚马逊信任的 CA 签署的服务器证书的HTTPS终端节点发送消息SNS。
8. 部署您创建的用于接收 Amazon SNS 消息的代码。当您订阅终端节点时，该终端节点必须准备好至少接收订阅确认消息。

第 2 步：使用HTTP/HTTPS终端节点订阅 Amazon SNS 主题

要通过主题向HTTP或HTTPS终端节点发送消息，您必须为终端节点订阅 Amazon SNS 主题。您可以使用终端节点来指定终端节点URL。要订阅主题，您可以使用亚马逊SNS控制台、sns-sub scribe 命令或 “订阅API” 操作。在开始之前，请确保您已URL准备好要订阅的终端节点，并且您的终端节点已准备好接收步骤 1 中所述的确认和通知消息。

使用 Amazon SNS 控制台为HTTP或HTTPS终端节点订阅主题

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 选择 Create subscription（创建订阅）。
4. 在 “协议” 下拉列表中，选择HTTP或HTTPS。
5. 在终端节点框中，粘贴您希望主题向其发送消息的终端节点的，然后选择创建订阅。URL
6. 将显示确认消息。选择关闭。

   将显示您的新订阅的订阅 ID PendingConfirmation。当您确认订阅时，Subscription ID (订阅 ID) 将显示订阅 ID。

第 3 步：确认您的亚马逊SNS订阅

要确认 A AWS mazon SNS 订阅，请按照以下步骤确保您的终端节点能够成功接收消息。此过程包括设置您的终端节点以处理传入的确认消息、检索必要的确认URL以及通过自动或手动方式确认订阅。

1. 订阅确认消息。在您为终端节点订阅亚马逊SNS主题后，Amazon SNS 会向该终端节点发送确认消息。此消息包含SubscribeURL，您需要用它来确认订阅。
2. 取回**SubscribeURL**. 您的终端节点应具有侦听和处理传入消息的代码。此代码必须SubscribeURL从确认消息中提取。确认消息通常以JSON有效载荷形式与SubscribeURL密钥一起到达。
3. 确认订阅。确认订阅的方法有两种：

   - 自动确认。您的终端节点代码可以自动访问SubscribeURL以确认订阅。这种方法要求您的终端节点向URL提供的端点发出HTTPGET请求。
   - 手动确认。如果未设置自动确认，则可以SubscribeURL使用 Web 浏览器手动访问。此步骤包括复制消息URL中的内容并将其粘贴到浏览器的地址栏中。

4. 验证订阅状态。通过访问确认订阅后SubscribeURL，Amazon SNS 会发送一份回复，其中包含一个带有名为的元素的XML文档SubscriptionArn。此元素包含订阅的 Amazon 资源名称 (ARN)，表示订阅已激活。
5. 使用亚马逊SNS控制台。您也可以使用验证订阅状态 AWS Management Console。导航至 Amazon SNS 控制面板，在 “订阅” 部分下找到您的订阅。已确认的订阅将显示其订阅ARN，而未确认的订阅将显示PendingConfirmation。

第 4 步：可选-为 Amazon SNS 订阅设置配送政策

默认情况下，如果消息的初始传送失败，Amazon 最多会SNS尝试三次重试，失败尝试之间的延迟设置为 20 秒。按照步骤 1 所述，您的终端节点应包括能够处理已重试消息的代码。通过为主题或订阅设置传送政策，您可以控制 Amazon SNS 重试失败消息的频率和间隔。您还可以在中指定 HTTP /S 通知的内容类型DeliveryPolicy。有关更多信息，请参阅 创建 HTTP /S 传输策略。

第 5 步：可选-向用户授予发布到 Amazon SNS 主题的权限

默认情况下，只有主题所有者才拥有发布主题的权限。要允许其他用户或应用程序向该主题发布内容，应使用 AWS Identity and Access Management (IAM) 授予该主题的发布权限。有关向IAM用户授予 Amazon SNS 操作权限的更多信息，请参阅在 Amazon 上使用基于身份的政策 SNS。

可以采取两种方式控制对主题的访问：

- 向IAM用户或组添加策略。向用户授予主题权限的最简单方式就是创建群组，向该群组添加适当策略，再向此群组添加用户。相比较而言，向群组添加或删除用户，比追踪您为单独用户而设定的各项策略要简单得多。
- 添加策略至主题。如果您想向另一 AWS 账户授予主题权限，那么唯一的方法是添加策略，并且该策略必须具备您想向其授予权限的主要 AWS 账户 。

绝大多数情况下，您应使用第一种方法（通过向群组添加或删除适当用户的方式，向群组添加策略，管理用户权限）。如果您需要向另一账户的用户授予权限，请使用第二种方法。

如果您向IAM用户或组添加了以下策略，则需要向该用户或群组中的成员授予对该主题执行sns:Publish操作的权限 MyTopic。

```
{
  "Statement":[{
    "Sid":"AllowPublishToMyTopic",
    "Effect":"Allow",
    "Action":"sns:Publish",
```

```
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

以下策略示例显示如何向主题授予另一账户权限。

> **Note**
>
> 当您向其他用户授予对您账户中某项资源的 AWS 账户 访问权限时，也就是向拥有该资源的管理员级访问权限（通配符访问权限）的IAM用户授予权限。其他账户中的所有其他IAM用户都会被自动拒绝访问您的资源。如果您想向该IAM用户授予 AWS 账户 访问您资源的权限，则该账户或具有管理员级别访问权限的IAM用户必须将资源权限委托给这些IAM用户。有关跨账户委托的更多信息，请参阅使用IAM指南中的启用跨账户访问权限。

如果您向账户 123456789012 MyTopic 中的主题添加了以下策略，则需要向账户 111122223333 授予对该主题执行操作的权限。`sns:Publish`

```
{
  "Statement":[{
    "Sid":"Allow-publish-to-topic",
    "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

步骤 6：将 Amazon SNS 消息发送到 HTTP /终HTTPS端节点

您可以通过发布到主题来向主题订阅发送消息。要向主题发布内容，您可以使用 Amazon SNS 控制台、`sns-publish`CLI命令或`Publish`API。

如果您遵循 步骤 1，则在您的终端节点上部署的代码将处理通知。

使用 Amazon SNS 控制台向主题发布内容

1. 使用 AWS 账户 或有权发布主题的IAM用户的证书，登录 AWS Management Console 并打开亚马逊SNS控制台，网址为https://console.aws.amazon.com/sns/。

2. 在导航面板上，选择主题，然后选择一个主题。
3. 选择发布消息按钮。
4. 在主题框中，输入主题（例如，**`Testing publish to my endpoint`**）。
5. 在消息框中，输入一些文本（例如 **`Hello world!`**），然后选择 发布消息。

   界面将显示如下消息：“Your message has been successfully published”（您的消息已成功发布）。

验证 Amazon SNS 消息的签名

要验证 Amazon 发送到您的HTTP终端节点的消息的真实性SNS，您可以验证消息签名。在两种情况下，我们建议验证消息的真实性。首先，当 Amazon SNS 向您的HTTP终端节点发送消息表示您订阅了某个主题时。其次，当 Amazon 在执行`Subscribe`或`Unsubscribe`API操作后SNS向您发送确认消息到您的HTTP终端节点时。

在验证 Amazon 发送的消息时，您应执行以下操作SNS：

- 从 Amazon 获取证书HTTPS时请务必使用SNS。
- 验证证书的真伪。
- 验证是否已收到来自亚马逊的证书SNS。
- 如果可能，请使用 Amazon 支持的 AWS SDKs方法之一SNS来验证和验证消息。
- 验证是否收到了您想要的 Amazon SNS 消息`TopicArn`。

Amazon SNS 支持两种消息签名版本：

- `SignatureVersion1`：Amazon 根据消息的SHA1哈希值SNS创建签名。
- `SignatureVersion2`：Amazon 根据消息的SHA256哈希值SNS创建签名。

在 Amazon SNS 主题上配置消息签名版本

默认情况下，Amazon SNS 主题使用 `SignatureVersion` 1。要在 Amazon SNS 主题上选择哈希算法（`SignatureVersion1` (SHA1) 或 `SignatureVersion` 2 (SHA256)），您可以使用`SetTopicAttributes`API操作。

以下代码示例显示如何使用 AWS CLI设置主题属性 `SignatureVersion`：

```
aws sns set-topic-attributes \
```

```
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

在使用HTTP基于查询的请求时验证 Amazon SNS 消息的签名

1. 从 Amazon SNS 发送到您的终端节点的HTTPPOST请求正JSON文中的文档中提取名称/值对。您将使用名称/值对中的一些值来创建待签字符串。在验证 Amazon SNS 消息的签名时，必须将转义的控制字符转换为其在Message和Subject值中的原始字符表示形式。当您将上述值用作待签字符串一部分时，上述值必须保留原始形式。有关如何解析JSON文档的信息，请参见第 1 步：确保您的终端节点已准备就绪，可以处理 Amazon SNS 消息。

   SignatureVersion告诉您 Amazon SNS 生成消息签名时使用的签名版本。通过签名版本，您可以确定生成签名的要求。对于通知，Amazon SNS 目前支持签名版本 1 和 2。本部分提供验证使用这些签名版本的签名的步骤。
2. 获取亚马逊SNS用来签署邮件的 X509 证书。指向 X509 证书位置的 SigningCertURL 值用于创建消息的数字签名。检索此位置上的证书。
3. 从此证书上提取公钥。来自 SigningCertURL 所指定证书的公钥用于验证信息的真实性和完整性。
4. 确定消息类型。待签字符串格式取决于消息类型，该类型由 Type 值指定。
5. 创建待签字符串。待签字符串为来自消息的特定名称-值对换行符逗号分隔列表。各个名称/值对由值后面换行符之后的第一个名称表示，以换行符为结尾。名称/值对必须以字节排序顺序予以列明。

   根据消息类型，待签字符串必须具有以下名称/值对。

   通知

   通知消息必须含有以下名称/值对：

   ```
   Message
   MessageId
   Subject (if included in the message)
   Timestamp
   TopicArn
   Type
   ```

   以下为针对 Notification 待签字符串的一个示例。

```
Message
My Test Message
MessageId
4d4dc071-ddbf-465d-bba8-08f81c89da64
Subject
My subject
Timestamp
2019-01-31T04:37:04.321Z
TopicArn
arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
Notification
```

SubscriptionConfirmation 和 UnsubscribeConfirmation

要验证SubscriptionConfirmation和UnsubscribeConfirmation消息的签名，待签字符串必须包含以下名称/值对：

```
Message
MessageId
SubscribeURL
Timestamp
Token
TopicArn
Type
```

以下示例是SubscriptionConfirmation消息的待签字符串：

```
Message
My Test Message
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
https://sns.us-east-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-2:123456789012:s4-
MySNSTopic-1G1WEFCOXTC0P&Token=233...
Timestamp
2019-01-31T19:25:13.719Z
Token
233...
TopicArn
```

```
arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
SubscriptionConfirmation
```

> **Note**
>
> 该SubscribeURL字段用于对UnsubscribeConfirmation消息SubscriptionConfirmation和消息进行签名的字符串。URL这是确认或管理订阅所必需的。

6. 通过 Base64 格式解码 Signature 值。消息传递以 Signature 值表示的签名，并将该签名编码为 Base64。将签名值与您计算出的签名进行对比前，应确保您已通过 Base64 完成对 Signature 值的解码操作，然后才能将运用相同格式的值进行对比。
7. 生成 Amazon SNS 消息的派生哈希值。使用与生成签名相同的哈希算法，以规范格式提交 Amazon SNS 消息。
   a. 如果SignatureVersion为 1，则SHA1用作哈希算法。
   b. 如果SignatureVersion为 2，则SHA256用作哈希算法。
8. 生成 Amazon SNS 消息的断言哈希值。断言的哈希值是使用公钥值（来自步骤 3）解密随着 Ama SNS zon 消息传送的签名的结果。
9. 验证 Amazon SNS 消息的真实性和完整性。比较派生的哈希值（来自步骤 7）与断言的哈希值（来自步骤 8）。如果值相同，则可以向接收者保证消息在传输过程中未被修改，并且消息必须来自亚马逊SNS。如果值不相同，则接收人不应信任它。

解析 Amazon SNS 消息格式

Amazon SNS 使用以下格式。

主题

- HTTP/HTTPS标题
- HTTP/HTTPS订阅确认JSON格式
- HTTP/HTTPS通知JSON格式
- HTTP/HTTPS取消订阅确认格式 JSON
- SetSubscriptionAttributes配送政策JSON格式

- SetTopicAttributes 配送政策JSON格式

HTTP/HTTPS标题

当亚马逊SNS向HTTP/HTTPS终端节点发送订阅确认、通知或取消订阅确认消息时，它会发送POST一条包含许多亚马逊SNS特定标头值的消息。您可以将标头值用于诸如识别消息类型之类的任务，而无需解析JSON消息正文即可读取该Type值。默认情况下，Amazon SNS 会将所有通知发送到Content-Type设置为 HTTP /S 的终端节点text/plain; charset=UTF-8。要选择除文本/纯文本（原定设置）以外的 Content-Type，请参阅创建 HTTP /S 传输策略中的 headerContentType。

x-amz-sns-message-type

消息类型。可能的值为 SubscriptionConfirmation、Notification 和 UnsubscribeConfirmation。

x-amz-sns-message-id

通用唯一标识符 (UUID)，对于发布的每封邮件都是唯一的。对于 Amazon SNS 在重试期间重新发送的通知，将使用原始消息的消息 ID。

x-amz-sns-topic-arn

此消息发布到的主题的 Amazon 资源名称 (ARN)。

x-amz-sns-subscription-arn

ARN用于订阅此终端节点。

以下HTTPPOST标头是发送到HTTP端点的Notification消息的标头示例。

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

HTTP/HTTPS订阅确认JSON格式

在您订阅HTTP/HTTPS终端节点后，Amazon SNS 会向HTTP/HTTPS终端节点发送订阅确认消息。此条消息包含您必须访问的 SubscribeURL 值，以确认订阅（或者，您可以将 Token 值与 ConfirmSubscription 结合使用）。

> **Note**
>
> 在确认订阅之前，Amazon SNS 不会向此终端节点发送通知

订阅确认消息是一条带有POST消息正文的消息，其中包含具有以下名称/值对的JSON文档。

Type

消息类型。为订阅确认，消息类型为：SubscriptionConfirmation。

MessageId

通用唯一标识符 (UUID)，对于发布的每封邮件都是唯一的。对于 Amazon SNS 在重试期间重新发送的消息，将使用原始消息的消息 ID。

Token

您可以使用 ConfirmSubscription 操作确认订阅的一个值。或者，您只需访问SubscribeURL。

TopicArn

此终端节点订阅的主题的 Amazon 资源名称 (ARN)。

Message

一个描述消息的字符串。为订阅确认，字符串看上去像这样：

```
You have chosen to subscribe to the topic arn:aws:sns:us-
east-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.
```

SubscribeURL

您URL必须访问才能确认订阅。或者，您可以改为将 Token 与 ConfirmSubscription 操作结合使用以确认订阅。

Timestamp

发送订阅确认的时间 (GMT)。

SignatureVersion

使用的亚马逊SNS签名版本。

- 如果 SignatureVersion 为 1，则 Signature 是 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 签名。
- 如果 SignatureVersion 为 2，则 Signature 是 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA256withRSA 签名。

Signature

Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 或 SHA256withRSA 签名。

SigningCertURL

用于URL对邮件进行签名的证书。

以下HTTPPOST消息是向HTTP终端节点发送SubscriptionConfirmation消息的示例。

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
```

```
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH
+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNKWLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

HTTP/HTTPS通知JSON格式

当 Amazon SNS 向订阅者HTTP或HTTPS终端节点发送通知时，发送到终端节点的消息正文包含具有以下名称/值对的JSON文档。POST

`Type`

消息类型。用于通知，这种类型属于`Notification`。

`MessageId`

通用唯一标识符 (UUID)，对于发布的每封邮件都是唯一的。对于 Amazon SNS 在重试期间重新发送的通知，将使用原始消息的消息 ID。

`TopicArn`

此消息发布到的主题的 Amazon 资源名称 (ARN)。

`Subject`

在将通知发布至主题时指定的 `Subject` 参数。

> **Note**
>
> 此参数为可选参数。如果未指`Subject`定，则此名称/值对不会出现在本JSON文档中。

`Message`

当通知发布至主题时指定的 `Message` 值。

`Timestamp`

发布通知的时间 (GMT)。

SignatureVersion

使用的亚马逊SNS签名版本。

- 如果 SignatureVersion 为 1，则 Signature 是 Message、MessageId、Subject（如果存在）、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 签名。
- 如果 SignatureVersion 为 2，则 Signature 是 Message、MessageId、Subject（如果存在）、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA256withRSA 签名。

Signature

Message、MessageId、Subject（如果存在）、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 或 SHA256withRSA 签名。

SigningCertURL

用于URL对邮件进行签名的证书。

UnsubscribeURL

A URL 可用于取消订阅此主题的终端节点。如果您访问此终端节点URL，Amazon 将SNS取消订阅终端节点并停止向该终端节点发送通知。

以下HTTPPOST消息是向HTTP终端节点发送Notification消息的示例。

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
```

```
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
  }
}
```

HTTP/HTTPS取消订阅确认格式 JSON

HTTP/HTTPS终端节点取消订阅主题后，Amazon 会向该终端节点SNS发送取消订阅确认消息。

取消订阅确认消息是一条POST消息，其消息正文包含具有以下名称 JSON /值对的文档。

`Type`

消息类型。为取消订阅确认，消息类型为`UnsubscribeConfirmation`。

`MessageId`

通用唯一标识符 (UUID)，对于发布的每封邮件都是唯一的。对于 Amazon SNS 在重试期间重新发送的消息，将使用原始消息的消息 ID。

`Token`

您可以使用 `ConfirmSubscription` 操作重新确认订阅的一个值。或者，您只需访问`SubscribeURL`。

`TopicArn`

此终端节点已取消订阅的主题的 Amazon 资源名称 (ARN)。

`Message`

一个描述消息的字符串。为了取消订阅确认，字符串应看起来像这样：

```
You have chosen to deactivate subscription arn:aws:sns:us-
east-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in this
 message.
```

SubscribeURL

您URL必须访问才能重新确认订阅。或者，您可以改为将 Token 与 ConfirmSubscription 操作结合使用以重新确认订阅。

Timestamp

发送取消订阅确认的时间 (GMT)。

SignatureVersion

使用的亚马逊SNS签名版本。

- 如果 SignatureVersion 为 1，则 Signature 是 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 签名。
- 如果 SignatureVersion 为 2，则 Signature 是 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA256withRSA 签名。

Signature

Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base64 编码 SHA1withRSA 或 SHA256withRSA 签名。

SigningCertURL

用于URL对邮件进行签名的证书。

以下HTTPPOST消息是向HTTP终端节点发送UnsubscribeConfirmation消息的示例。

```
POST / HTTP/1.1
x-amz-sns-message-type: UnsubscribeConfirmation
x-amz-sns-message-id: 47138184-6831-46b8-8f7c-afc488602d7d
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1399
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

```
{
  "Type" : "UnsubscribeConfirmation",
  "MessageId" : "47138184-6831-46b8-8f7c-afc488602d7d",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to deactivate subscription arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in this
 message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37fb6...",
  "Timestamp" : "2012-04-26T20:06:41.581Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEHXgJm...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

SetSubscriptionAttributes配送政策JSON格式

如果您向SetSubscriptionAttributes操作发送请求并将AttributeName参数的值设置为DeliveryPolicy，则该AttributeValue参数的值必须是有效的JSON对象。例如，以下例子将传输策略设置为 5 次重试。

```
http://sns.us-east-2.amazonaws.com/
?Action=SetSubscriptionAttributes
&SubscriptionArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
%3A80289ba6-0fd4-4079-afb4-ce8c8260f0ca
&AttributeName=DeliveryPolicy
&AttributeValue={"healthyRetryPolicy":{"numRetries":5}}
...
```

使用以下JSON格式作为AttributeValue参数的值。

```
{
    "healthyRetryPolicy" : {
        "minDelayTarget" :  int,
        "maxDelayTarget" : int,
        "numRetries" : int,
        "numMaxDelayRetries" : int,
        "backoffFunction" : "linear|arithmetic|geometric|exponential"
```

```
    },
    "throttlePolicy" : {
        "maxReceivesPerSecond" : int
    },
    "requestPolicy" : {
        "headerContentType" : "text/plain | application/json | application/xml"
    }
}
```

有关该SetSubscriptionAttribute操作的更多信息，请SetSubscriptionAttributes访问《亚马逊简单通知服务API参考》。有关支持的HTTP内容类型标头的更多信息，请参阅。创建 HTTP /S 传输策略

SetTopicAttributes 配送政策JSON格式

如果您向SetTopicAttributes操作发送请求并将AttributeName参数的值设置为DeliveryPolicy，则该AttributeValue参数的值必须是有效的JSON对象。例如，以下例子将传输策略设置为 5 次重试。

```
http://sns.us-east-2.amazonaws.com/
?Action=SetTopicAttributes
&TopicArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
&AttributeName=DeliveryPolicy
&AttributeValue={"http":{"defaultHealthyRetryPolicy":{"numRetries":5}}}
...
```

使用以下JSON格式作为AttributeValue参数的值。

```
{
    "http" : {
        "defaultHealthyRetryPolicy" : {
            "minDelayTarget":  int,
            "maxDelayTarget": int,
            "numRetries": int,
            "numMaxDelayRetries": int,
            "backoffFunction": "linear|arithmetic|geometric|exponential"
        },
        "disableSubscriptionOverrides" : Boolean,
        "defaultThrottlePolicy" : {
            "maxReceivesPerSecond" : int
        },
        "defaultRequestPolicy" : {
            "headerContentType" : "text/plain | application/json | application/xml"
```

```
        }
    }
}
```

有关该SetTopicAttribute操作的更多信息，请SetTopicAttributes访问《亚马逊简单通知服务API参考》。有关支持的HTTP内容类型标头的更多信息，请参阅。创建 HTTP /S 传输策略

Fanout Amazon SNS 事件到 AWS 事件分叉管道

对于事件存档和分析，亚马逊SNS现在建议使用其与Amazon Data Firehose的原生集成。您可以将 Firehose 传输流订阅SNS主题，这样您就可以向存档和分析终端节点发送通知，例如亚马逊简单存储服务 (Amazon S3) 存储桶、Amazon Redshift 表、 OpenSearch 亚马逊OpenSearch 服务（服务）等。将 Amazon SNS 与 Firehose 传输流配合使用是一种完全托管且无需代码的解决方案，您无需使用任何功能。 AWS Lambda 有关更多信息，请参阅 Fanout Amazon SNS 通知，带有 Firehose 传送流，可增强数据管理。

您可以使用 Amazon SNS 构建事件驱动的应用程序，这些应用程序使用订阅服务自动执行工作，以响应发布商服务触发的事件。此架构模式可提高服务的可重用性、可互操作性和可扩展性。但是，将事件处理分解为可满足常见事件处理要求的管道（例如，事件存储、备份、搜索、分析和重放）可能会非常耗费人力。

为了加快事件驱动型应用程序的开发，您可以将事件处理管道（由事件分叉管道提供支持）订阅到 Amazon 主题。 AWS SNS AWS Event Fork Pipelines 是一套基于AWS 无服务器应用程序模型 (AWS SAM) 的开源嵌套应用程序，您可以直接从 Ev AWS ent Fork Pipelines 套件（选择显示创建自定义 IAM角色或资源策略的应用程序）将其部署到您的 AWS 账户中。

有关 E AWS vent Fork Pipelines 用例，请参阅部署和测试 Amazon SNS 事件分叉管道示例应用程序。

主题

- AWS 事件分叉管道的工作原理
- 部署 AWS 事件分叉管道
- 部署和测试 Amazon SNS 事件分叉管道示例应用程序
- 将 AWS 事件分叉管道订阅到 Amazon 主题 SNS

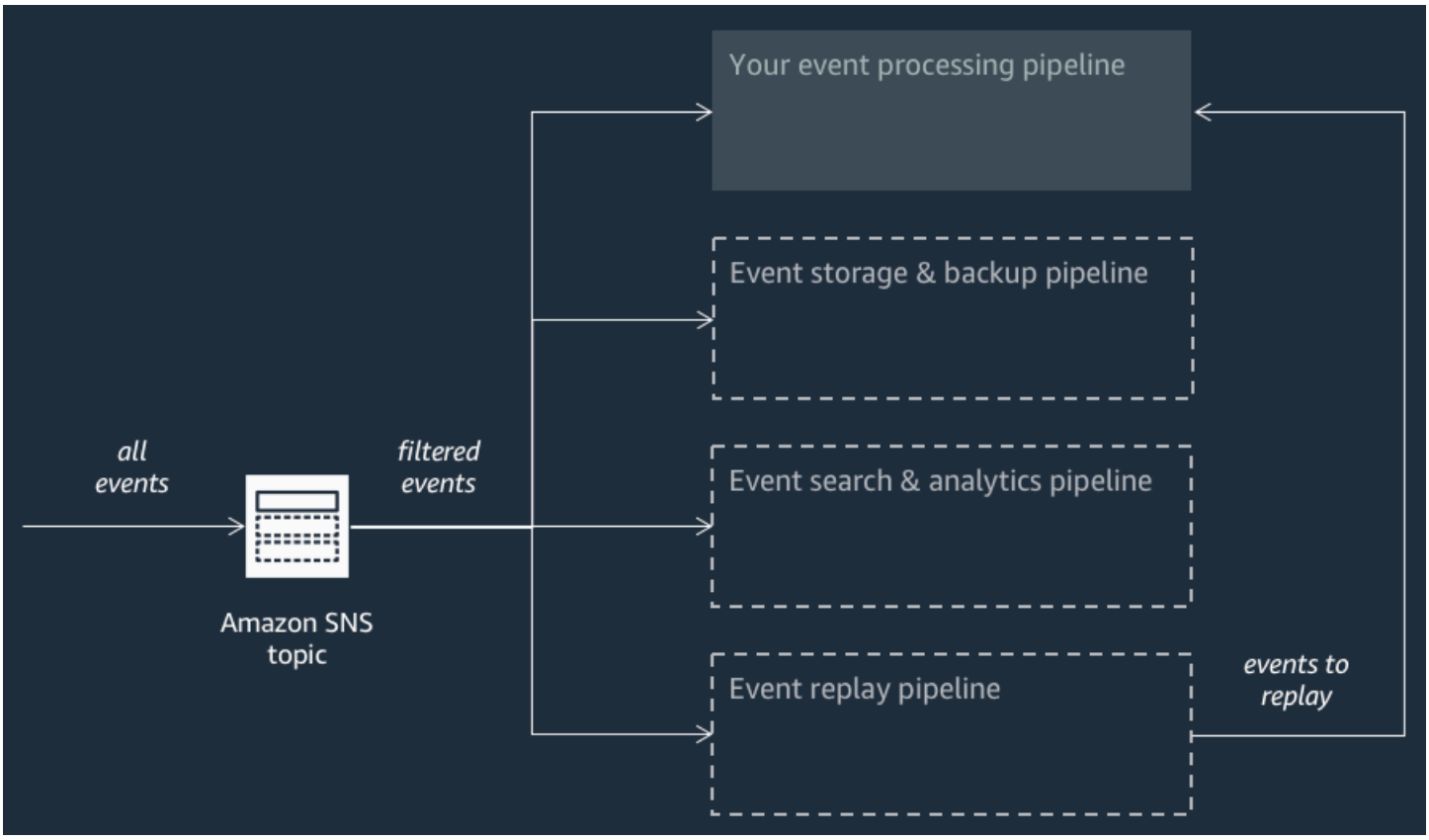
AWS 事件分叉管道的工作原理

AWS Event Fork Pipelines 是一种无服务器设计模式。但是，它也是一套基于嵌套的无服务器应用程序 AWS SAM（您可以直接从 AWS Serverless Application Repository (AWS SAR) 将其部署到您的，AWS 账户 以丰富您的事件驱动平台）。您可以根据架构的需要单独部署这些嵌套的应用程序。

主题

- 事件存储与备份管道
- 事件搜索与分析管道
- 事件重播管道

下图显示了一个由三个嵌套应用程序补充的 AWS Event Fork Pipelines 应用程序。您可以根据架构的要求在上 AWS SAR独立部署 E AWS vent Fork Pipelines 套件中的任何管道。



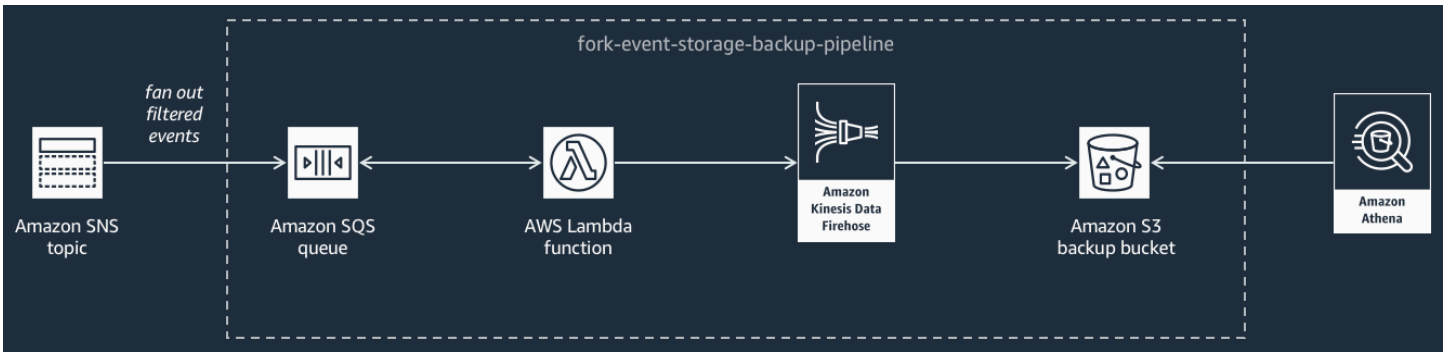
每个管道都订阅了相同的 Amazon SNS 主题，从而允许自己在将事件发布到该主题时并行处理事件。每个管道都是独立的，并且可以设置其自己的订阅筛选策略。这允许管道仅处理它感兴趣的部分事件（而不是发布到主题的所有事件）。

> **Note**
>
> 由于您将三个 AWS 事件分叉管道放置在常规事件处理管道旁边（可能已经订阅了您的 Amazon SNS 主题），因此您无需更改当前消息发布者的任何部分即可在现有工作负载中利用 AWS 事件分叉管道。

事件存储与备份管道

下图显示了事件存储与备份管道。您可以将此管道订阅到您的 Amazon SNS 主题，以自动备份流经您系统的事件。

该管道由一个用于缓冲亚马逊SNS主题传送的事件的 Amazon SQS 队列、一个自动轮询队列中这些事件并将其推送到 Amazon Data Firehose 流的 AWS Lambda 函数，以及一个持久备份流加载的事件的 Amazon S3 存储桶。

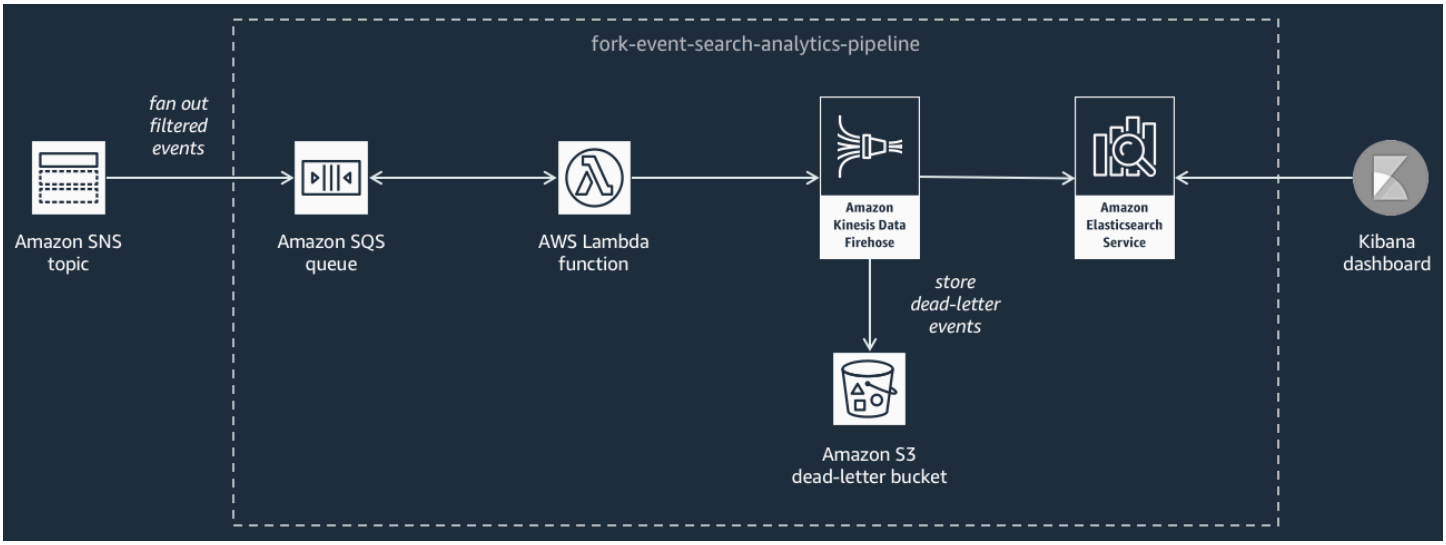


要微调 Firehose 流的行为，可将其配置为在将事件加载到存储桶之前对事件进行缓冲、转换和压缩。加载事件后，您可以使用 Amazon Athena 使用标准查询来查询存储桶。SQL您也可以将管道配置为重用现有 Amazon S3 存储桶或创建一个新的存储桶。

事件搜索与分析管道

下图显示了事件搜索与分析管道。您可以将此管道订阅到您的 Amazon SNS 主题，以便在搜索域中将流经您系统的事件编入索引，然后对其进行分析。

该管道由一个用于缓冲亚马逊SNS主题传送的事件的亚马逊SQS队列、一个轮询队列中的事件并将其推送到亚马逊数据 Firehose 流的 AWS Lambda 函数、一个用于索引 Firehose 流加载的事件的 OpenSearch 亚马逊服务域以及一个存储无法在搜索域中编入索引的死信事件的 Amazon S3 存储桶。



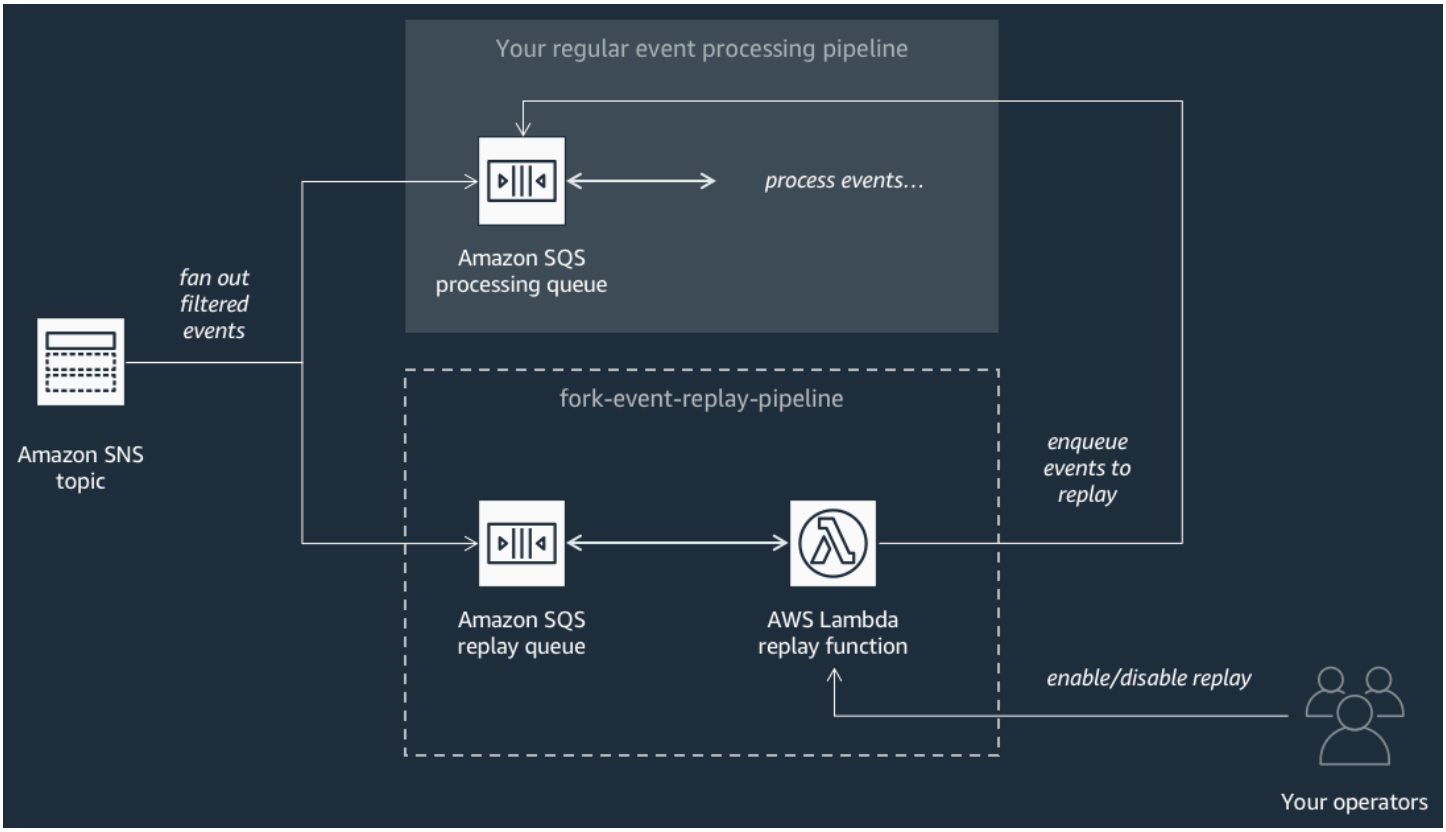
要在事件缓冲、转换和压缩方面微调 Firehose 流，您可以配置此管道。

您还可以配置管道是应重复使用您的现有 OpenSearch 域 AWS 账户 还是为您创建一个新域。在搜索域中为事件编制索引时，您可以使用 Kibana 对事件运行分析并实时更新可视化控制面板。

事件重播管道

下图显示了事件重播管道。要记录过去 14 天内您的系统处理的事件（例如，当您的平台需要从故障中恢复时），您可以将此管道订阅您的 Amazon SNS 主题，然后重新处理这些事件。

该管道由一个 Amazon SQS 队列组成，该队列用于缓冲由 Amazon SNS 主题传送的事件，以及一个用于轮询队列中的事件并将其重新驱动到您的常规事件处理管道的 AWS Lambda 函数，该管道也订阅了您的主题。



> **Note**
>
> 默认情况下，重播功能已禁用，而不会重新导入您的事件。如果您需要重新处理事件，则必须启用 Amazon SQS 重播队列作为重 AWS Lambda 播功能的事件源。

部署 AWS 事件分叉管道

AWS Event Fork Pipelines 套件（选择 “显示创建自定义IAM角色或资源策略的应用程序”）在中作为一组公共应用程序提供 AWS Serverless Application Repository，您可以从中使用AWS Lambda 控制台手动部署和测试它们。有关使用 AWS Lambda 控制台部署管道的信息，请参阅将 AWS 事件分叉管道订阅到 Amazon 主题 SNS。

在生产场景中，我们建议在整个应用程序的 AWS SAM模板中嵌入 AWS Event Fork Pipelines。嵌套应用程序功能允许您通过向 AWS SAM模板添加资源、引用嵌套应用程序`SemanticVersion`的 AWS SAR`ApplicationId`和`AWS::Serverless::Application`来实现此目的。

例如，您可以将事件存储和备份管道用作嵌套应用程序，方法是将以下YAML代码段添加到 AWS SAM模板的`Resources`部分。

```
Backup:
  Type: AWS::Serverless::Application
  Properties:
    Location:
      ApplicationId: arn:aws:serverlessrepo:us-east-2:123456789012:applications/fork-
event-storage-backup-pipeline
      SemanticVersion: 1.0.0
    Parameters:
      #The ARN of the Amazon SNS topic whose messages should be backed up to the Amazon
 S3 bucket.
      TopicArn: !Ref MySNSTopic
```

指定参数值时，您可以使用 AWS CloudFormation 内部函数来引用模板中的其他资源。例如，在上面的YAML片段中，`TopicArn`参数引用了 AWS SAM 模板中其他地方定义的`AWS::SNS::Topic`资源`MySNSTopic`。有关更多信息，请参阅 AWS CloudFormation 用户指南中的内置函数参考。

> **Note**
>
> AWS SAR应用程序的 AWS Lambda 控制台页面包括 “复制为SAM资源” 按钮，该按钮可将嵌套 AWS SAR应用程序YAML所需的内容复制到剪贴板。

部署和测试 Amazon SNS 事件分叉管道示例应用程序

为了加快事件驱动型应用程序的开发，您可以将事件处理管道（由事件分叉管道提供支持）订阅到 Amazon 主题。 AWS SNS AWS Event Fork Pipelines 是一套基于AWS 无服务器应用程序模型 (AWS SAM) 的开源嵌套应用程序，您可以直接从 Ev AWS ent Fork Pipelines 套件（选择显示创建自定义 IAM角色或资源策略的应用程序）将其部署到您的 AWS 账户中。有关更多信息，请参阅 AWS 事件分叉管道的工作原理。

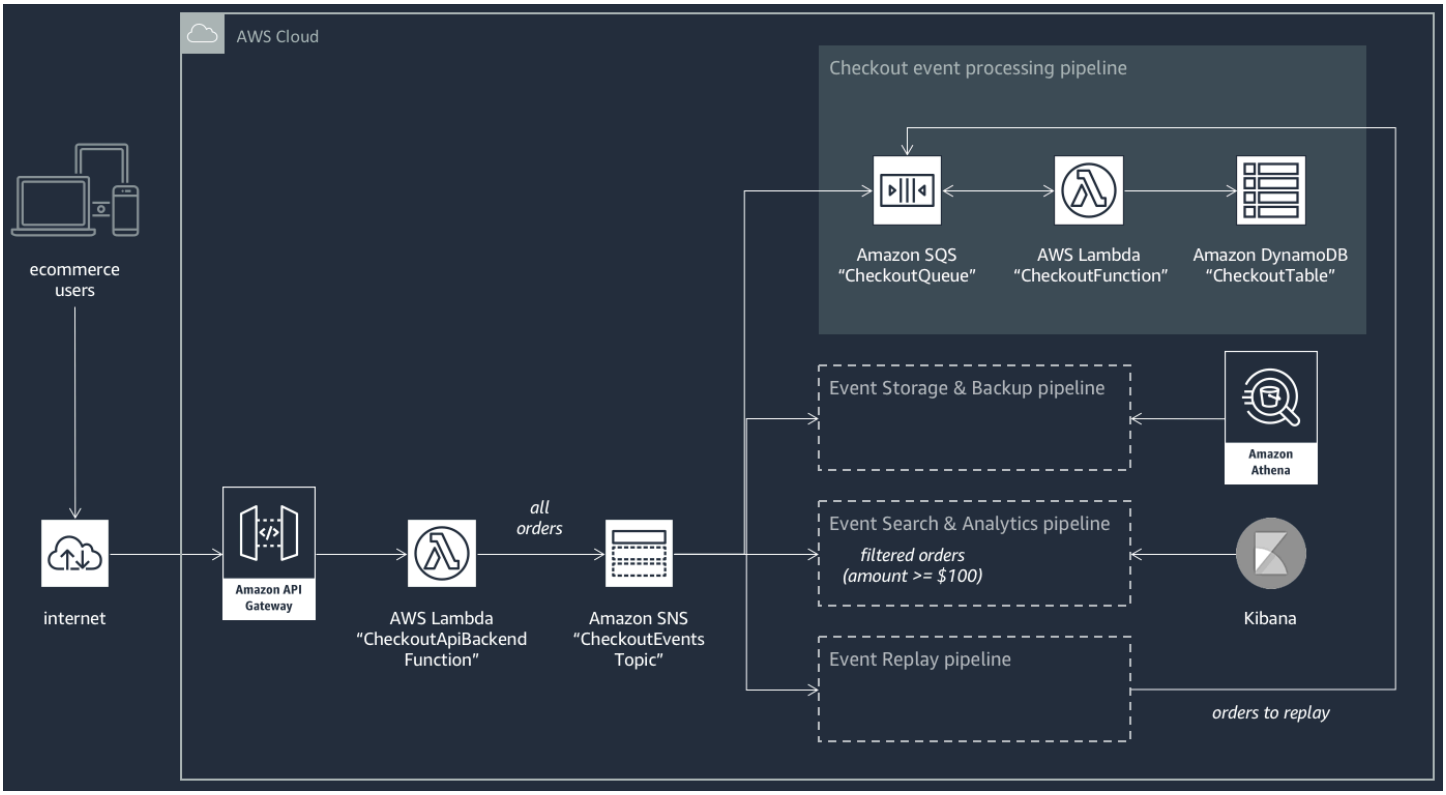
本页介绍如何使用部署和测试 E AWS vent Fork Pipelines 示例应用程序。 AWS Management Console

> **Important**
>
> 为避免在部署完 AWS 事件分支管道示例应用程序后产生不必要的费用，请删除其 AWS CloudFormation 堆栈。有关更多信息，请参阅 AWS CloudFormation 用户指南中的在 AWS CloudFormation 控制台上删除堆栈。

主题

AWS 事件分叉管道用例示例

以下场景描述了一个使用 Event Fork Pipelines AWS 的事件驱动型无服务器电子商务应用程序。您可以在中使用此示例电子商务应用程序， AWS Serverless Application Repository 然后 AWS 账户使用控制台将其部署到您的 AWS Lambda 控制台中，您可以在其中对其进行测试并检查其源代码 GitHub。



该电子商务应用程序通过Gate API way RESTful API 托管并由该 AWS Lambda 功能提供支持的买家订单`CheckoutApiBackendFunction`。此函数将所有收到的订单发布到名为的 Amazon SNS 主题，`CheckoutEventsTopic`该主题反过来又会将订单分散到四个不同的渠道。

第一个管道是由电子商务应用程序的拥有者设计和实现的常规结算处理管道。该管道具有用于缓冲所有已收到订单的 Amazon SQS 队列`CheckoutQueue`、一个名为`CheckoutFunction`的 AWS Lambda 函数（用于轮询队列以处理这些订单）以及用于安全保存所有已下订单的 DynamoDB `CheckoutTable` 表。

应用 AWS 事件分叉管道

电子商务应用程序的组件处理核心业务逻辑。但是，电子商务应用程序拥有者还需满足：

- 合规性 - 安全的、压缩的静态加密备份，清理敏感信息
- 弹性 - 在执行过程中断的情况下重播最近的订单
- 可搜索性 - 对已下订单运行分析并生成指标

应用程序所有者无需实现此事件处理逻辑，而是可以订阅 E AWS vent Fork Pipelines 到 `CheckoutEventsTopic` Amazon SNS 主题

- 事件存储与备份管道配置为转换数据以删除信用卡信息、缓冲数据 60 秒、使用数据进行压缩GZIP，然后使用 Amazon S3 的默认客户托管密钥对其进行加密。此密钥由 () 管理 AWS 并由 AWS Key Management Service (AWS KMS) 提供支持。

  有关更多信息，请参阅《亚马逊数据 Firehose 开发者指南》中的 “为目的地选择亚马逊 S3”、“亚马逊数据 Firehos e 数据转换” 和 “配置设置”。
- 为事件搜索与分析管道配置了一个 30 秒的索引重试持续时间、一个用于存储无法在搜索域中编制索引的订单的存储桶和一个用来限制已编制索引的订单集的筛选策略。

  有关更多信息，请参阅 Amazon Data Firehose 开发者指南中的为您的目的地选择 OpenSearch 服务。
- 事件重播管道使用由电子商务应用程序所有者设计和实施的常规订单处理管道的 Amazon SQS 队列部分进行配置。

  有关更多信息，请参阅队列名称和 URL Amazon 简单队列服务开发者指南。

以下JSON筛选策略是在事件搜索和分析管道的配置中设置的。它仅匹配总金额为 100 美元或更多的传入订单。有关更多信息，请参阅 Amazon SNS 邮件过滤。

```
{
   "amount": [{ "numeric": [ ">=", 100 ] }]
}
```

使用 E AWS vent Fork Pipelines 模式，电子商务应用程序所有者可以避免开发开销，这种开销通常是在为事件处理编写非微分逻辑之后出现的。相反，她可以直接将 AWS 事件分叉管道部署 AWS Serverless Application Repository 到她身上 AWS 账户。

步骤 1：部署示例 Amazon SNS 应用程序

1. 登录 AWS Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)，然后选择 Create function (创建函数)。
3. 在 Create function (创建函数) 页面上，执行以下操作：
   a. 选择 “浏览无服务器应用程序存储库”、“公共应用程序”、“显示创建自定义IAM角色或资源策略的应用程序”。
   b. 搜索 `fork-example-ecommerce-checkout-api`，然后选择该应用程序。
4. 在 fork-example-ecommerce-checkout-api 页面上，执行以下操作：
   a. 在 Application settings (应用程序设置) 部分中，输入 Application name (应用程序名称)（例如，`fork-example-ecommerce-my-app`）。

      > **Note**
      > - 要稍后轻松找到您的资源，请保留前缀 `fork-example-ecommerce`。
      > - 对于每个部署，应用程序名称必须唯一。如果您重复使用应用程序名称，则部署将仅更新先前部署的 AWS CloudFormation 堆栈（而不是创建新的堆栈）。

   b. （可选）输入以下LogLevel设置之一以执行应用程序的 Lambda 函数：
      - `DEBUG`
      - `ERROR`
      - `INFO`（默认值）
      - `WARNING`
5. 选择我确认此应用程序创建自定义IAM角色、资源策略并部署嵌套应用程序。 然后在页面底部选择 “部署”。

关于 fork-example-ecommerce-的部署状态***my-app***页面上，Lambda 会显示 “您的应用程序正在部署中” 状态。

在 “资源” 部分中， AWS CloudFormation 开始创建堆栈并显示每个资源的 CREATE_IN_ PROGRESS 状态。该过程完成后， AWS CloudFormation 将显示 CREATE_ COMPLETE 状态。

> **Note**
>
> 部署所有资源可能需要 20-30 分钟。

部署完成后，Lambda 将显示 Your application has been deployed（您的应用程序已部署完成）状态。

步骤 2：执行SNS链接的示例应用程序

1. 在 AWS Lambda 控制台的导航面板上，选择应用程序。
2. 在 Applications (应用程序) 页面上的搜索字段中，搜索 `serverlessrepo-fork-example-ecommerce-`*`my-app`*，然后选择该应用程序。
3. 在 Resources (资源) 部分中，执行以下操作：
   a. 例如，要查找类型为的资源 ApiGatewayRestApi，请按类型对资源进行排序`ServerlessRestApi`，然后展开该资源。
   b. 将显示两个嵌套资源，分别是 “ApiGateway部署” 和 “ApiGateway阶段”。
   c. 复制链接 Prod API 端点并附加到`/checkout`该端点，例如：

   ```
   https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
   ```

4. 将以下内容复制JSON到名为的文件中`test_event.json`。

```
{
   "id": 15311,
   "date": "2019-03-25T23:41:11-08:00",
   "status": "confirmed",
   "customer": {
      "id": 65144,
      "name": "John Doe",
      "email": "john.doe@example.com"
   },
   "payment": {
      "id": 2509,
      "amount": 450.00,
```

```
      "currency": "usd",
      "method": "credit",
      "card-network": "visa",
      "card-number": "1234 5678 9012 3456",
      "card-expiry": "10/2022",
      "card-owner": "John Doe",
      "card-cvv": "123"
   },
   "shipping": {
      "id": 7600,
      "time": 2,
      "unit": "days",
      "method": "courier"
   },
   "items": [{
      "id": 6512,
      "product": 8711,
      "name": "Hockey Jersey - Large",
      "quantity": 1,
      "price": 400.00,
      "subtotal": 400.00
   }, {
      "id": 9954,
      "product": 7600,
      "name": "Hockey Puck",
      "quantity": 2,
      "price": 25.00,
      "subtotal": 50.00
   }]
}
```

5. 要向您的API终端节点发送HTTPS请求，请通过执行curl命令将示例事件负载作为输入传递，例如：

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

API返回以下空响应，表示成功执行：

```
{ }
```

第 3 步：验证 Amazon SNS 应用程序和管道性能

步骤 1：验证示例结账管道的执行情况

1. 登录 Amazon DynamoDB 控制台。
2. 在导航面板上，选择 Tables (表)。
3. 搜索 `serverlessrepo-fork-example` 并选择 `CheckoutTable`。
4. 在表详细信息页面上，选择 Items (项目)，然后选择已创建的项目。

   将显示存储的属性。

步骤 2：验证事件存储和备份管道的执行情况

1. 登录 Amazon S3 控制台。
2. 在导航面板上，选择 Buckets (存储桶)。
3. 搜索 `serverlessrepo-fork-example`，然后选择 `CheckoutBucket`。
4. 导航目录层次结构，直到找到扩展名为 `.gz` 的文件。
5. 要下载该文件，请依次选择 Actions (操作) 和 Open (打开)。
6. 为管道配置了一个 Lambda 函数，此函数将清理信用卡信息以实现合规性。

   要验证存储的JSON有效载荷是否不包含任何信用卡信息，请解压缩该文件。

步骤 3：验证事件搜索和分析管道的执行情况

1. 登录到OpenSearch 服务控制台。
2. 在导航面板上的 My domains (我的域) 下，选择前缀为 `serverl-analyt` 的域。
3. 该管道配置了设置数字匹配条件的 Amazon SNS 订阅筛选策略。

   要验证该事件是否已被索引，因为它指的是价值高于 USD 100 美元的订单，请在 serverl-analyt-***abcdefgh1ijk***页面上，选择指数，checkout _events。

步骤 4：验证事件重播管道的执行情况

1. 登录 Amazon SQS 控制台。
2. 在队列列表中，搜索 `serverlessrepo-fork-example` 并选择 `ReplayQueue`。

3. 选择发送和接收消息。
4. 在 “发送和接收消息” 中 fork-example-ecommerce-***my-app***... replayP--ReplayQueue***123ABCD4E5F6***对话框中，选择 “轮询留言”。
5. 要验证事件是否已入队，请选择队列中显示的消息旁边的 More Details (更多详细信息)。

步骤 4：模拟问题并重播事件以进行恢复

步骤 1：启用模拟问题并发送第二个API请求

1. 登录 AWS Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)。
3. 搜索 `serverlessrepo-fork-example` 并选择 `CheckoutFunction`。
4. 在 fork-example-ecommerce-***my-app***-CheckoutFunction-***ABCDEF***... 页面的环境变量部分中，将 BUG_ ENABLED 变量设置为 true，然后选择保存。
5. 将以下内容复制JSON到名为的文件中`test_event_2.json`。

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
    "customer": {
       "id": 56999,
       "name": "Marcia Oliveira",
       "email": "marcia.oliveira@example.com"
    },
    "payment": {
       "id": 3311,
       "amount": 75.00,
       "currency": "usd",
       "method": "credit",
       "card-network": "mastercard",
       "card-number": "1234 5678 9012 3456",
       "card-expiry": "12/2025",
       "card-owner": "Marcia Oliveira",
       "card-cvv": "321"
    },
    "shipping": {
       "id": 9900,
       "time": 20,
       "unit": "days",
```

```
        "method": "plane"
    },
    "items": [{
        "id": 9993,
        "product": 3120,
        "name": "Hockey Stick",
        "quantity": 1,
        "price": 75.00,
        "subtotal": 75.00
    }]
}
```

6. 要向您的API终端节点发送HTTPS请求，请通过执行curl命令将示例事件负载作为输入传递，例如：

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

API返回以下空响应，表示成功执行：

```
{ }
```

步骤 2：验证模拟数据损坏

1. 登录 Amazon DynamoDB 控制台。
2. 在导航面板上，选择 Tables (表)。
3. 搜索 `serverlessrepo-fork-example` 并选择 `CheckoutTable`。
4. 在表详细信息页面上，选择 Items (项目)，然后选择已创建的项目。

将显示存储的属性，其中一些标记为 CORRUPTED!

步骤 3：禁用模拟问题

1. 登录 AWS Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)。
3. 搜索 `serverlessrepo-fork-example` 并选择 `CheckoutFunction`。
4. 在 fork-example-ecommerce-***my-app***-CheckoutFunction-***ABCDEF***... 页面的环境变量部分中，将 BUG_ ENABLED 变量设置为 false，然后选择保存。

步骤 4：启用重播以从问题中恢复

1. 在 AWS Lambda 控制台的导航面板上，选择功能。
2. 搜索 `serverlessrepo-fork-example` 并选择 `ReplayFunction`。
3. 展开 “设计器” 部分，选择SQS磁贴，然后在该SQS部分中选择 “启用”。

> **Note**
>
> 启用 Amazon SQS 事件源触发器大约需要 1 分钟。

4. 选择保存。
5. 要查看已恢复的属性，请返回到 Amazon DynamoDB 控制台。
6. 要禁用重播，请返回 AWS Lambda 控制台并禁用 Amazon SQS 事件源触发器`ReplayFunction`。

将 AWS 事件分叉管道订阅到 Amazon 主题 SNS

为了加快事件驱动型应用程序的开发，您可以将事件处理管道（由事件分叉管道提供支持）订阅到 Amazon 主题。 AWS SNS AWS Event Fork Pipelines 是一套基于AWS 无服务器应用程序模型 (AWS SAM) 的开源嵌套应用程序，您可以直接从 Ev AWS ent Fork Pipelines 套件（选择显示创建自定义 IAM角色或资源策略的应用程序）将其部署到您的 AWS 账户中。有关更多信息，请参阅 AWS 事件分叉管道的工作原理。

本节介绍如何使用部署管道，然后 AWS Management Console 为 AWS 事件分支管道订阅 Amazon SNS 主题。在开始之前，请创建一个 Amazon SNS 主题。

要删除构成管道的资源，请在 AWS Lambda 控制台的应用程序页面上找到管道，展开SAM模板部分，选择CloudFormation堆栈，然后选择其他操作，删除堆栈。

主题

- 向 Amazon 部署和订阅事件存储和备份管道 SNS
- 部署事件搜索和分析管道并将其订阅到 Amazon SNS
- 使用 Amazon SNS 集成部署事件重播管道

向 Amazon 部署和订阅事件存储和备份管道 SNS

对于事件存档和分析，亚马逊SNS现在建议使用其与Amazon Data Firehose的原生集成。您可以将 Firehose 传输流订阅SNS主题，这样您就可以向存档和分析终端节点发送通知，例如亚马逊简单存储服务 (Amazon S3) 存储桶、Amazon Redshift 表、 OpenSearch 亚马逊OpenSearch 服务（服务）等。将 Amazon SNS 与 Firehose 传输流配合使用是一种完全托管且无需代码的解决方案，您无需使用任何功能。 AWS Lambda 有关更多信息，请参阅 Fanout Amazon SNS 通知，带有 Firehose 传送流，可增强数据管理。

本页介绍如何部署事件存储和备份管道以及如何将其订阅 Amazon SNS 主题。此过程会自动将与管道关联的 AWS SAM 模板转换为 AWS CloudFormation 堆栈，然后将该堆栈部署到您的 AWS 账户。此过程还会创建和配置构成事件存储与备份管道的资源集，包括以下内容：

- 亚马逊SQS队列
- Lambda 函数
- Firehose 传输流
- Amazon S3 备份存储桶

有关配置以 S3 存储桶作为目标的流的更多信息，请参阅 Amazon Data Firehos API e 参考S3DestinationConfiguration中的。

有关转换事件以及配置事件缓冲、事件压缩和事件加密的更多信息，请参阅《Amazon Data Fire hose 开发者指南》中的创建亚马逊数据 Firehose 传输流。

有关筛选事件的更多信息，请参阅本指南中的 亚马逊SNS订阅筛选政策。

1. 登录 AWS Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)，然后选择 Create function (创建函数)。
3. 在 Create function (创建函数) 页面上，执行以下操作：
   a. 选择 “浏览无服务器应用程序存储库”、“公共应用程序”、“显示创建自定义IAM角色或资源策略的应用程序”。
   b. 搜索 `fork-event-storage-backup-pipeline`，然后选择该应用程序。
4. 在 fork-event-storage-backup-pipelin e 页面上，执行以下操作：

a. 在 Application settings (应用程序设置) 部分中，输入 Application name (应用程序名称) (例如，my-app-backup)。

> **Note**
>
> - 对于每个部署，应用程序名称必须唯一。如果您重复使用应用程序名称，则部署将仅更新先前部署的 AWS CloudFormation 堆栈（而不是创建新的堆栈）。

b. （可选）对于 BucketArn，请输入加载传入事件的 S3 存储桶的。ARN如果您未输入值，则会在您的 AWS 账户中创建一个新的 S3 存储桶。

c. （可选）对于 DataTransformationFunctionArn，输入用于转换传入事件的 Lambda 函数。ARN如果您不输入值，则将禁用数据转换。

d. （可选）输入以下LogLevel设置之一以执行应用程序的 Lambda 函数：

- DEBUG
- ERROR
- INFO (默认值)
- WARNING

e. 对于 TopicArn，ARN输入要订阅此分叉管道实例的 Amazon SNS 主题。

f. （可选）对于StreamBufferingIntervalInSeconds和 StreamBufferingSizeInMBs，输入用于配置传入事件缓冲的值。如果您不输入任何值，则使用 300 秒和 5 MB。

g. （可选）输入以下StreamCompressionFormat设置之一以压缩传入的事件：

- GZIP
- SNAPPY
- UNCOMPRESSED (默认值)
- ZIP

h. （可选）对于 StreamPrefix，输入字符串前缀以命名存储在 S3 备份存储桶中的文件。如果您不输入值，则不使用任何前缀。

i. （可选）对于 SubscriptionFilterPolicy，请按JSON格式输入用于筛选传入事件的 Amazon SNS 订阅筛选政策。筛选策略决定在 OpenSearch 服务索引中对哪些事件进行索引。如果您不输入值，则不使用筛选（为所有事件编制索引）。

j. （可选）对于 SubscriptionFilterPolicyScope，输入字符串MessageBody或MessageAttributes以启用基于负载或基于属性的邮件筛选。

k. 选择我确认此应用程序创建自定义IAM角色、资源策略并部署嵌套应用程序。 然后选择 “部署”。

关于的部署状态 ***my-app***页面上，Lambda 会显示 “您的应用程序正在部署中” 状态。

在资源部分中， AWS CloudFormation 开始创建堆栈并显示每个资源的 CREATE_IN_ PROGRESS 状态。该过程完成后， AWS CloudFormation 将显示 CREATE_ COMPLETE 状态。

部署完成后，Lambda 将显示 Your application has been deployed（您的应用程序已部署完成）状态。

发布到您的 Amazon SNS 主题的消息存储在事件存储和备份管道自动配置的 S3 备份存储桶中。

部署事件搜索和分析管道并将其订阅到 Amazon SNS

对于事件存档和分析，亚马逊SNS现在建议使用其与Amazon Data Firehose的原生集成。您可以将 Firehose 传输流订阅SNS主题，这样您就可以向存档和分析终端节点发送通知，例如亚马逊简单存储服务 (Amazon S3) 存储桶、Amazon Redshift 表、 OpenSearch 亚马逊OpenSearch 服务（服务）等。将 Amazon SNS 与 Firehose 传输流配合使用是一种完全托管且无需代码的解决方案，您无需使用任何功能。 AWS Lambda 有关更多信息，请参阅 Fanout Amazon SNS 通知，带有 Firehose 传送流，可增强数据管理。

本页介绍如何部署事件搜索和分析管道以及如何将其订阅 Amazon SNS 主题。此过程会自动将与管道关联的 AWS SAM 模板转换为 AWS CloudFormation 堆栈，然后将该堆栈部署到您的 AWS 账户。此过程还会创建和配置构成事件搜索与分析管道的资源集，包括以下内容：

- 亚马逊SQS队列
- Lambda 函数
- Firehose 传输流
- 亚马逊 OpenSearch 服务域名
- Amazon S3 死信存储桶

有关配置以索引为目标的流的更多信息，请参阅`ElasticsearchDestinationConfiguration`《Amazon Data Firehos API e 参考》。

有关转换事件以及配置事件缓冲、事件压缩和事件加密的更多信息，请参阅《Amazon Data Fire hose 开发者指南》中的创建亚马逊数据 Firehose 传输流。

有关筛选事件的更多信息，请参阅本指南中的 亚马逊SNS订阅筛选政策。

1. 登录 AWS Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)，然后选择 Create function (创建函数)。
3. 在 Create function (创建函数) 页面上，执行以下操作：
   a. 选择 “浏览无服务器应用程序存储库”、“公共应用程序”、“显示创建自定义IAM角色或资源策略的应用程序”。
   b. 搜索 `fork-event-search-analytics-pipeline`，然后选择该应用程序。
4. 在 fork-event-search-analytics-pipelin e 页面上，执行以下操作：
   a. 在 Application settings (应用程序设置) 部分中，输入 Application name (应用程序名称)（例如，`my-app-search`）。

      > **Note**
      > 对于每个部署，应用程序名称必须唯一。如果您重复使用应用程序名称，则部署将仅更新先前部署的 AWS CloudFormation 堆栈（而不是创建新的堆栈）。

   b. （可选）对于 DataTransformationFunctionArn，请输入用于转换传入事件的 Lambda 函数。ARN如果您不输入值，则将禁用数据转换。
   c. （可选）输入以下LogLevel设置之一以执行应用程序的 Lambda 函数：
      - `DEBUG`
      - `ERROR`
      - `INFO`（默认值）
      - `WARNING`
   d. （可选）对于 OpenSearch 服务域 SearchDomainArn，输入配置所需计算和存储功能的集群。ARN如果您不输入值，则使用默认配置创建新域。
   e. 对于 TopicArn，ARN输入要订阅此分叉管道实例的 Amazon SNS 主题。
   f. 对于 SearchIndexName，输入用于事件搜索和分析的 OpenSearch 服务索引的名称。

> **Note**
>
> 以下配额适用于索引名称：
>
> - 不能包含大写字母
> - 不能包含以下字符：\ / * ? " < > | ` , #
> - 不能以下列字符开头：- + _
> - 不能为以下内容：. ..
> - 长度不能超过 80 个字符
> - 长度不能超过 255 个字节
> - 不能包含冒号（来自 OpenSearch 服务 7.0）

g. （可选）为 OpenSearch 服务索引的轮换周期输入以下SearchIndexRotationPeriod设置之一：

- `NoRotation`（默认值）
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

索引轮换将时间戳附加到索引名称，从而促进旧数据的到期。

h. 对于 SearchTypeName，输入用于在索引中组织事件的 OpenSearch 服务类型的名称。

> **Note**
>
> - OpenSearch 服务类型名称可以包含任何字符（空字节除外），但不能以开头_。
> - 对于 S OpenSearch ervice 6.x，每个索引只能有一个类型。如果您为已有其他类型的现有索引指定新类型，Firehose 将返回运行时错误。

i. （可选）对于StreamBufferingIntervalInSeconds和 StreamBufferingSizeInMBs，输入用于配置传入事件缓冲的值。如果您不输入任何值，则使用 300 秒和 5 MB。

j. （可选）输入以下StreamCompressionFormat设置之一以压缩传入的事件：

- GZIP
- SNAPPY
- UNCOMPRESSED（默认值）
- ZIP

k. （可选）对于 StreamPrefix，输入字符串前缀以命名存储在 S3 死信存储桶中的文件。如果您不输入值，则不使用任何前缀。

l. （可选）对于 StreamRetryDurationInSecons，输入 Firehose 无法索引服务索引中的 OpenSearch 事件时的重试持续时间。如果您不输入值，则使用 300 秒。

m. （可选）对于 SubscriptionFilterPolicy，请按JSON格式输入用于筛选传入事件的 Amazon SNS 订阅筛选政策。筛选策略决定在 OpenSearch 服务索引中对哪些事件进行索引。如果您不输入值，则不使用筛选（为所有事件编制索引）。

n. 选择我确认此应用程序创建自定义IAM角色、资源策略并部署嵌套应用程序。 然后选择 “部署”。

关于的部署状态 ***my-app-search***页面上，Lambda 会显示 “您的应用程序正在部署中” 状态。

在资源部分中， AWS CloudFormation 开始创建堆栈并显示每个资源的 CREATE_IN_ PROGRESS 状态。该过程完成后， AWS CloudFormation 将显示 CREATE_ COMPLETE 状态。

部署完成后，Lambda 将显示 Your application has been deployed（您的应用程序已部署完成）状态。

发布到您的 Amazon SNS 主题的消息将在事件搜索和分析管道自动配置的 OpenSearch 服务索引中编制索引。如果该管道无法为事件编制索引，它会将其存储在 S3 死信存储桶中。

使用 Amazon SNS 集成部署事件重播管道

本页介绍如何部署事件重播管道以及如何将其订阅 Amazon SNS 主题。此过程会自动将与管道关联的 AWS SAM 模板转换为 AWS CloudFormation 堆栈，然后将该堆栈部署到您的 AWS 账户。此过程还会创建和配置构成事件重播管道的一组资源，包括 Amazon SQS 队列和 Lambda 函数。

有关筛选事件的更多信息，请参阅本指南中的 亚马逊SNS订阅筛选政策。

1. 登录 AWS Lambda 控制台。
2. 在导航面板上，选择 Functions (函数)，然后选择 Create function (创建函数)。
3. 在 Create function (创建函数) 页面上，执行以下操作：

a. 选择 “浏览无服务器应用程序存储库”、“公共应用程序”、“显示创建自定义IAM角色或资源策略的应用程序”。

b. 搜索 `fork-event-replay-pipeline`，然后选择该应用程序。

4. 在该fork-event-replay-pipeline页面上，执行以下操作：

a. 在 Application settings (应用程序设置) 部分中，输入 Application name (应用程序名称)（例如，`my-app-replay`）。

> **Note**
>
> 对于每个部署，应用程序名称必须唯一。如果您重复使用应用程序名称，则部署将仅更新先前部署的 AWS CloudFormation 堆栈（而不是创建新的堆栈）。

b. （可选）输入以下LogLevel设置之一以执行应用程序的 Lambda 函数：

- `DEBUG`
- `ERROR`
- `INFO`（默认值）
- `WARNING`

c. （可选）对于 ReplayQueueRetentionPeriodInSeconds，输入 Amazon SQS 重播队列保留消息的时间长度（以秒为单位）。如果您不输入值，则使用 1209600 秒（14 天）。

d. 对于 TopicArn，ARN输入要订阅此分叉管道实例的 Amazon SNS 主题。

e. 对于 DestinationQueueName，输入 Lambda 重播函数将消息转发到的亚马逊SQS队列的名称。

f. （可选）对于 SubscriptionFilterPolicy，请按JSON格式输入用于筛选传入事件的 Amazon SNS 订阅筛选政策。筛选策略决定缓冲哪些事件以进行重播。如果您不输入值，则不使用筛选（缓冲所有事件以进行重播）。

g. 选择我确认此应用程序创建自定义IAM角色、资源策略并部署嵌套应用程序。 然后选择 “部署”。

关于的部署状态 ***my-app-replay***页面上，Lambda 会显示 “您的应用程序正在部署中” 状态。

在资源部分中， AWS CloudFormation 开始创建堆栈并显示每个资源的 CREATE_IN_ PROGRESS 状态。该过程完成后， AWS CloudFormation 将显示 CREATE_ COMPLETE 状态。

部署完成后，Lambda 将显示 Your application has been deployed（您的应用程序已部署完成）状态。

发布到您的 Amazon SNS 主题的消息将在事件重播管道自动配置的 Amazon SQS 队列中进行缓冲以供重播。

> **Note**
>
> 默认情况下，禁用重播。要启用重播，请导航到 Lambda 控制台上的函数页面，展开 “设计器” 部分，选择SQS磁贴，然后在该SQS部分中选择 “启用”。

在亚马逊上使用 Amazon EventBridge 排程器 SNS

Amazon Sched EventBridge uler 是一种无服务器计划程序，允许您通过一个中央托管服务创建、运行和管理任务。使用 EventBridge Scheduler，您可以使用 Cron 和费率表达式为重复模式创建计划，也可以配置一次性调用。您可以设置灵活的交付时间窗口，定义重试限制，并为失败的API调用设置最大保留时间。

本页介绍如何使用 EventBridge 计划程序按计划发布来自亚马逊SNS主题的消息。

主题

设置执行角色

创建新计划时， EventBridge 调度器必须有权代表您调用其目标API操作。您可以使用执行角色向 EventBridge 调度器授予这些权限。您附加到计划执行角色的权限策略定义了所需权限。这些权限取决于API您希望 EventBridge 调度程序调用的目标。

当您使用 EventBridge 调度器控制台创建计划时，如以下过程所示， EventBridge 调度器会根据您选择的目标自动设置执行角色。如果要使用 EventBridge 调度器SDKs、或中的一个来创建计划 AWS CLI AWS CloudFormation，则必须有一个现有的执行角色来授予 EventBridge 调度器调用目标所需的权限。有关为计划手动设置执行角色的更多信息，请参阅《日程EventBridge 安排器用户指南》中的设置执行角色。

创建计划

使用控制台创建计划

1. 在https://console.aws.amazon.com/scheduler/家中打开 Amazon EventBridge 日程安排器控制台。
2. 在计划页面，选择创建计划。
3. 在指定计划详细信息页面，在计划名称和描述部分中，执行以下操作：

   a. 对于计划名称，输入计划的名称。例如，**`MyTestSchedule`**。

   b. （可选）对于描述，输入对计划的描述。例如，**`My first schedule`**。

   c. 对于计划组，从下拉列表中选择一个计划组。如果您没有计划组，选择默认。要创建计划组，选择创建自己的计划。

      您可以使用计划组将标签添加到计划组。

4. • 选择计划选项。

出现	请执行此操作...
一次性计划 一次性计划仅在您指定的日期和时间调用一次目标。	对于日期和时间，请执行以下操作： • 输入 YYYY/MM/DD 格式的有效日期。 • 输入 24 小时 hh:mm 格式的时间戳。 • 对于时区，选择时区。
定期计划 定期计划按照您使用 cron 表达式或 rate 表达式指定的速率调用目标。	a. 对于计划类型，执行以下操作之一： • 要使用 cron 表达式定义计划，请选择基于 cron 的计划并输入 cron 表达式。 • 要使用 rate 表达式定义计划，请选择基于

出现	请执行此操作...	
	rate 的计划并输入 rate 表达式。 有关 cron 和费率表达式的更多信息，请参阅 A mazon EventBridge 计划程序用户指南中的计划 EventBridge 程序中的计划类型。 b. 对于灵活的时间窗口，选择关闭以关闭该选项，或者选择一个预定义的时间窗口。例如，如果您选择 15 分钟并且将定期计划设置为每小时调用一次其目标，则该计划将在每小时开始后的 15 分钟内运行。	

5. （可选）如果您在上一步中选择定期计划，在时间范围部分，请执行以下操作：
   a. 对于时区，请选择时区。
   b. 对于开始日期和时间，请输入 YYYY/MM/DD 格式的有效日期，然后指定 24 小时 hh:mm 格式的时间戳。
   c. 对于结束日期和时间，请输入 YYYY/MM/DD 格式的有效日期，然后指定 24 小时 hh:mm 格式的时间戳。
6. 选择下一步。
7. 在 “选择目标” 页上，选择 EventBridge 调度器调用的 AWS API操作：
   a. 选择 Amazon P SNS ublish。
   b. 在发布部分中，选择 SNS 主题或选择创建新的 SNS 主题。
   c. （可选）输入 JSON 负载。如果您未输入有效负载，则 EventBridge 调度器会使用空事件来调用该函数。
8. 选择下一步。

9. 在 Settings (设置) 页面上，执行以下操作：

   a. 要打开计划，在计划状态下，切换启用计划。

   b. 要为您的计划配置重试策略，请在 “重试策略” 和 “死信队列 (DLQ)” 下执行以下操作：

      - 切换重试。
      - 对于事件的最大持续时间，请输入 EventBridge 调度器必须保留未处理事件的最大小时数和最小值。
      - 最长时间为 24 小时。
      - 在 “最大重试次数” 中，输入目标返回错误时 EventBridge 调度器重试计划的最大次数。

        最大值为 185 次重试。

      使用重试策略，如果调 EventBridge 度未能调用其目标，则调度程序会重新运行该计划。如果已配置，则必须为计划设置最长保留时间和最大重试次数。

   c. 选择 EventBridge 日程安排器存储未传送事件的位置。

死信队列 () 选项 DLQ	请执行此操作...
请勿存储	选择 None。
将活动存储 AWS 账户 在您创建日程安排的同一位置	a. 在 “我 AWS 账户 的” 中选择 “选择一个 Amazon SQS 队列” DLQ。 b. 选择亚马逊SQS队列的亚马逊资源名称 (ARN)。
将活动存储在与您创建日程表不同的 AWS 账户 位置	a. 选择将 “其他” 中的 Amazon SQS 队列指定 AWS 账户 为DLQ。 b. 输入亚马逊SQS队列的亚马逊资源名称 (ARN)。

   d. 要使用客户托管密钥加密目标输入，在加密下，选择自定义加密设置（高级）。

如果选择此选项，请输入现有KMS密钥ARN或选择创建 AWS KMS key以导航到 AWS KMS 控制台。有关 EventBridge 计划程序如何加密静态数据的更多信息，请参阅 A mazon EventBridge 计划程序用户指南中的静态加密。

e. 要让 EventBridge Scheduler 为您创建新的执行角色，请选择为此计划创建新角色。然后，在角色名称中输入名称。如果您选择此选项，S EventBridge cheduler 会将模板化目标所需的权限附加到该角色。

10. 选择下一步。
11. 在查看并创建计划页面上，查看计划的详细信息。在每个部分中，选择编辑返回到该步骤并编辑其详细信息。
12. 选择创建计划。

    您可以在计划页面上查看新的和现有的计划列表。在状态列下，验证新计划是否已启用。

相关资源

有关 EventBridge 调度程序的更多信息，请参阅以下内容：

- EventBridge 日程安排器用户指南
- EventBridge 调度器参考 API
- EventBridge 调度程序定价

使用 Amaz SNS on application-to-person 发送消息

本节提供有关使用 Amazon SNS 向订阅者发送用户通知的信息，例如移动应用程序、手机号码和电子邮件地址。

主题

通过 Amazon 发送移动短信 SNS

您可以使用 Amazon SNS 向支持该SMS功能的设备发送短信或SMS消息。您可以直接向电话号码发送消息，也可以使用多个电话号码订阅主题，然后通过向该主题发送消息来一次向这些电话号码发送消息。

您可以为 AWS 账户设置SMS首选项，以根据您的用例和预算量身定制SMS配送。例如，您可以选择是否在成本和可靠传输方面优化消息。您还可以为各个消息传输指定支出配额以及为您的 AWS 账户指定每月支出配额。

在当地法律和法规（例如美国和加拿大）要求的情况下，SMS收件人可以选择退出，这意味着他们选择停止接收来自您的SMS邮件 AWS 账户。收件人退出后，您可以在某些限制条件下重新加入该电话号码，以便继续向其发送消息。

Amazon SNS 支持SMS在多个地区发送消息，您可以向 200 多个国家和地区发送消息。有关更多信息，请参阅 Amazon SNS 支持的国家和地区。

主题

- Amazon SNS 支持的国家和地区
- 亚马逊SNSSMS最佳实践

Amazon SNS SMS 沙箱

当您开始使用 Amazon SNS 发送SMS消息时，您的 AWS 账户处于SMS沙箱中。SMS沙盒为您提供了一个安全的环境，让您可以试用 Amazon 的SNS功能，而不必冒发件人声誉的SMS风险。当您的账户处于SMS沙箱状态时，您可以使用 Amazon 的所有功能SNS，但有以下限制：

- 您只能向经过验证的目标电话号码发送SMS消息。
- 您最多可以有 10 个已验证的目标电话号码。
- 您只能在验证或上次验证尝试后 24 小时或更长时间内删除目标电话号码。

当您的账户移出沙箱后，这些限制就会被移除，您可以向任何收件人发送SMS消息。

主题

- 在 Amazon SNS SMS 沙箱中添加和验证电话号码
- 从 Amazon SNS SMS 沙箱中删除电话号码
- 退出 Amazon SNS SMS 沙箱

在 Amazon SNS SMS 沙箱中添加和验证电话号码

要开始在 AWS 账户处于SMS沙箱状态时发送SMS消息，请创建发起人身份，添加目标电话号码，然后对其进行验证。

> **Note**
>
> 与不在SMS沙箱中的账户一样，在向某些国家或地区的收件人发送SMS消息之前，需要提供原始身份。有关更多信息，请参阅 Amazon SNS 支持的国家和地区。
> 发件人IDs包括发件IDs人和不同类型的发件人号码。要查看您现有的发货号，请在亚马逊SNS控制台的导航窗格中选择原产地编号。目前，发件人IDs未出现在此列表中。

要添加和验证目标电话号码

1. 登录 Amazon SNS 控制台。

2. 为电话号码创建原始身份。
3. 在控制台菜单中，选择支持消息SMS传递的AWS 区域。
4. 在导航窗格中，选择短信 (SMS)。
5. 在移动短信 (SMS) 页面的沙盒目标电话号码下，选择添加电话号码。
6. 在 Destination details（目标详细信息）下，输入国家/地区代码和电话号码，指定要用于验证消息的语言，然后选择 Add phone number（添加电话号码）。

   Amazon SNS 向目标电话号码发送一次性密码 (OTP)。如果目标电话号码OTP在 15 分钟内未收到验证码，请选择 “重新发送验证码”。您最多可以每 24 小时OTP向同一个目的地电话号码发送五次。
7. 在验证码框中，输入OTP发送到目标的电话号码，然后选择验证电话号码。

   目标电话号码及其验证状态显示在 Sandbox destination phone numbers（沙盒目标电话号码）部分。如果验证状态为 Pending（待处理），验证未成功。例如，如果您没有为电话号码输入国家/地区代码，就会发生这种情况。您只能在验证或上次验证尝试后 24 小时或更长时间内删除待处理或已验证的目标电话号码。
8. 在要使用此目标电话号码的每个区域中，重复以上步骤。

未收到短信疑难解答 OTP

解决可能导致电话号码无法接收OTP短信的常见问题。

- Amazon SNS SMS 支出限额：如果您 AWS 账户 已超过发送SMS消息的支出上限，则在提高限额或解决账单问题之前，可能无法发送更多消息（包括OTP短信）。
- 未选择接收SMS通知的电话号码：在某些国家或地区，收件人必须选择接收来自短码的SMS消息，短代码通常用于短OTP信。如果收件人的电话号码未被选中，他们将不会收到OTP短信。
- 运营商限制或过滤：某些移动运营商可能设置了限制或过滤机制，以阻止某些类型的SMS消息（包括短OTP信）的传送。这可能是由于运营商实施的安全政策或反垃圾邮件措施所致。
- 电话号码无效或不正确：如果收件人提供的电话号码不正确或无效，则不会发送OTP短信。
- 网络问题：临时的网络问题或中断可能会导致无法向收件人的手机SMS发送消息（包括OTP短信）。
- 延迟传送：在某些情况下，由于网络拥塞或其他因素，SMS邮件的传送可能会出现延迟。案OTP文最终可能会交付，但可能会延迟到预期的时间范围之外。
- 账户暂停或终止：如果您的账户存在问题 AWS 账户，例如未付款或违反服务 AWS 条款，则可能会暂停或终止包括OTP短信在内的亚马逊SNS消息功能。

从 Amazon SNS SMS 沙箱中删除电话号码

您可以从SMS沙箱中删除待处理或已验证的目标电话号码。

从SMS沙箱中删除目标电话号码

1. 在验证电话号码后等待 24 小时，或者在您最后一次验证尝试后等待 24 小时。
2. 登录 Amazon SNS 控制台。
3. 在控制台菜单中，选择一个支持SMS发送消息的AWS 区域，并在其中添加了目标电话号码。
4. 在导航窗格中，选择短信 (SMS)。
5. 在移动短信 (SMS) 页面的沙盒目标电话号码下，选择要删除的电话号码，然后选择删除电话号码。
6. 要确认您要删除电话号码，请输入 **`delete me`**，然后选择 Delete（删除）。

   如果您验证或尝试验证目标电话号码后已过去 24 小时或更长时间，则该号码将被删除，Amazon会SNS更新您的目标电话号码列表。
7. 在添加目标电话号码但不再计划使用它的每个区域中重复这些步骤。

退出 Amazon SNS SMS 沙箱

将您移 AWS 账户 出SMS沙箱需要您先添加、验证和测试目标电话号码。然后，您必须使用创建案例 AWS Support。

请求将您的 AWS 账户移出SMS沙箱

1. 验证电话号码

   a. 当您在SMS沙盒中 AWS 账户 时，打开 Amazon SNS 控制台。

   b. 在导航窗格的 “移动” 下，选择 “短信” (SMS)。

   c. 在沙盒目标电话号码部分，添加并验证一个或多个目标电话号码。此验证可确保您可以成功发送和接收消息。
2. 测试SMS发布

   - 确认您能够向至少一个经过验证的电话号码发送和接收消息。有关如何发布SMS消息的更多详细说明，请参阅使用 Amazon 向手机发布SMS消息 SNS。
3. 启动沙箱编辑

- 在 Amazon SNS 控制台的移动短信 (SMS) 页面上，在 “账户信息” 下，选择 “退出SMS沙箱”。此操作会将您重定向到 Amazon Support Center，并在选择服务配额增加选项的情况下自动创建支持案例。

4. 填写表格

   - 在服务配额增加下的支持表单中，执行以下操作：

     i. 选择选择SNS短信作为服务。

     ii. 提供您打算从中发送SMS消息的网站URL或应用程序名称。

     iii. 指定您要发送的消息类型：一次性密码、促销消息或交易消息。

     iv. 选择您要AWS 区域从中发送SMS消息的。

     v. 列出您计划向其发送SMS消息的国家或地区。

     vi. 描述您的客户如何选择接收消息。

     vii. 包括您打算使用的任何消息模板。

5. 指定配额和区域

   - 在 Requests (请求) 下，执行以下操作：

     i. 选择你要移动AWS 区域的地方 AWS 账户。

     ii. 为 “资源类型” 选择 “一般限制”。

     iii. 为配额选择 “退出SMS沙箱”。

     iv. （可选）要申请额外加薪或其他调整，请选择添加其他请求并指定必要的详细信息。

     v. 在 “新配额值” 栏中输入USD您请求的限额。

6. 其他细节

   a. 在问题描述中，提供与您的请求相关的所有其他详细信息。

   b. 在 “联系人选项” 下，选择您的首选联系语言。

7. 提交请求

   - 选择 “提交”，将您的请求发送至 AWS Support。

AWS Support 团队会在 24 小时内对您的请求做出初步回应。

为了防止我们的系统被用于发送未经请求或恶意的内容，我们要仔细考虑每个请求。如果我们可以，我们将在 24 小时内准予您的请求。但是，如果我们需要从您那里获得其他信息，则可能需要更长的时间来解决您的请求。

如果您的使用案例与我们的策略不符，我们可能无法准予您的请求。

Amazon SNS SMS 消息的来源身份

当您使用 Amazon 发送SMS消息时SNS，您可以使用以下类型的原始身份向收件人表明自己的身份：

- Amazon IDs 中的发件人 SNS
- 了解 Amazon 中的原产地编号 SNS

> **Note**
>
> 亚马逊SNSSMS消息在目前不支持 Amazon Pinpoint 的地区可用。如果您在欧洲（斯德哥尔摩）、中东（巴林）、欧洲（巴黎）、南美洲（圣保罗）或美国西部（加利福尼亚北部）开展业务，请在美国东部（弗吉尼亚北部）地区打开 Amazon Pinpoint 控制台注册您的 10 DLC 家公司和活动，但不要申请 10 DLC 号码。相反，请使用 S AWS ervice Quotas 控制台创建服务限制提高案例，同时请求该区域的 10 DLC 数字。要了解有关如何请求源身份的更多信息，请参阅 为使用 Amazon SNS 进行 SMS 消息收发请求支持。

Amazon IDs 中的发件人 SNS

发件人 ID 是一个字母名称，用于标识SMS邮件的发件人。当您使用发件人 ID 发送SMS消息，并且收件人位于支持发件人 ID 身份验证的区域时，您的发件人 ID 会显示在收件人的设备上，而不是电话号码上。与电话号码、长代码或短代码相比，发件人 ID 为SMS收件人提供的有关发件人的信息更多。

全球多个国家和地区IDs都支持发件人。在某些地方，如果您是一家向个人客户发送SMS消息的企业，则必须使用在监管机构或行业团体中预先注册的发件人 ID。有关支持或要求发件人的国家和地区的完整列表IDs，请参阅Amazon SNS 支持的国家和地区。

使用发件人不收取额外费用IDs。但是，对于发件人 ID 身份验证的支持和要求各不相同。一些主要市场（包括加拿大、中国和美国）不支持使用发件人IDs。某些地区要求向个人客户发送SMS消息的公司必须使用预先在监管机构或行业团体注册的发件人 ID。

> **Important**
>
> AWS 禁止SMS欺骗，即使用发件人 ID 冒充他人、公司或产品。仅使用代表您拥有的品牌或商标的发件人 ID。

优点

发件人IDs向收件人提供有关邮件发件人的更多信息。使用发件人 ID 比使用长代码或短代码更容易建立您的品牌标识。使用发件人 ID 无需额外付费。

劣势

各个国家或地区对于发件人 ID 身份验证的支持和要求并不一致。在几个主要市场 (包括加拿大、中国和美国)，不支持发件人 ID。在某些地区，您必须先获得监管机构的IDs预先批准才能使用您的发件人。

按国家/地区划分的发送人 ID 注册

您需要向 AWS 支持人员提交案例才能注册发送SMS消息IDs的发件人。提出支持案例后， AWS 将分享其他必需的文件。您还必须提供您在其中注册发件人 ID 的相应国家/地区的以下信息。

国家/地区名称	消息类型	格式限制和要求	注册要求
澳大利亚（AU）	交易和促销	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符 • 发件人 ID 必须是发件公司的品牌名称 SMS	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量

国家/地区名称	消息类型	格式限制和要求	注册要求

			• 使用案例说明，消息的用途
			• 如果不明显，请解释公司名称和发送人 ID 之间的联系
			• 公司的官方营业/贸易许可证号或 # VAT
			• 您计划发送的消息模板
			• 商业登记许可证。示例包括但不限于：
			  • 澳大利亚商业号码 (ABN)
			  • 澳大利亚公司编号 (ACN)
			  • 澳大利亚注册机构编号 (ARBN)
			  • 原住民公司编号 (ICN)
			• 授权书 (LOA)

国家/地区名称	消息类型	格式限制和要求	注册要求
白俄罗斯（BY）	仅事务消息	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符 • 发件人 ID 必须是发件公司的品牌名称 SMS	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 公司的官方营业/贸易许可证号或 # VAT • 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
中国（CN） 中国不要求发送人 ID 注册，但确实要求注册内容/模板以及正文前附签名（例如，[Amazon]）。	不允许使用以下关键字： • 法轮功 • SB • 天安门广场 归属以下类别的消息： • 信用卡 • 数字支付（包括加密货币） • 欺诈流量URLs（网络钓鱼或垃圾邮件） • 赌博 • 不当内容（成人、暴力、毒品、酒精） • 贷款 • 整形外科 • 政治 • 宗教 • 股票交易 • 虚拟货币 SMS邮件正文前面必须加上方括号内的签名。要成功发送到中国，您需要随消息内容模板注册消息签	• 最多 11 个字符 • 没有空格 • 无特殊字符	• 公司名称 • 公司地址 • 州/省 • Country • 公司电话号码 • 公司网站 • 每月估计的发送量 • 消息类型：促销/事务 • 使用案例解释 • 您要注册的模板（SMS发送的模板不得偏离为确保合规性和成功交付而提供的模板）。例如，[CompanyName] 您的一次性密码是 {OTP}。此代码将在 10 分钟后过期。 • 确认您不会将促销内容作为事务消息类型发送 • 消息签名。请参阅签名格式限制和要求。

国家/地区名称	消息类型	格式限制和要求	注册要求
	名。每次SMS发送的消息内容都必须加上此签名。不在SMS邮件正文前面加上注册的签名可能会导致被屏蔽或过滤SMS。正文前面必须加上方括号SMS内的签名。		

国家/地区名称	消息类型	格式限制和要求	注册要求
埃及（EG）	仅事务消息	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 您的公司在埃及是否有商业实体/办事处？ 或者，这是一家向埃及发送消息的非埃及公司吗？ • 书面确认此使用案例涵盖此发送人 ID 发送的所有消息 • 如果不明显，请提供公司名称和发送人 ID 之间的联系

国家/地区名称	消息类型	格式限制和要求	注册要求
			• 您计划发送的消息模板
印度（IN）	仅事务消息	• 字母数字 • 最多 6 个字符 • 没有空格 • 无特殊字符	请参阅 印度的 Amazon SNS 发件人 ID 注册要求。

国家/地区名称	消息类型	格式限制和要求	注册要求
约旦（JO）	仅事务消息	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 商业登记证书 • 您计划发送的消息类型（例如OTP，提醒） • 书面确认此使用案例描述了该发送人 ID 要发送的所有消息 • 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
科威特（KW）	仅事务消息	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 您计划发送的消息类型（例如OTP，提醒） • 书面确认此使用案例描述了该发送人 ID 要发送的所有消息 • 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
菲律宾（PH）	仅事务消息	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 您计划发送的消息类型（例如OTP，提醒） • 书面确认此使用案例描述了该发送人 ID 要发送的所有消息 • 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
卡塔尔（QA）	仅事务消息	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 发送SMS的类型 • （Transactional/Promotional/OTP）请注意，为了合规，只能将交易/ OTP 消息用于发往卡塔尔的发件人IDs。 • 书面确认此使用案例描述了该发送人 ID 要发送的所有消息

国家/地区名称	消息类型	格式限制和要求	注册要求
			• 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
俄罗斯（RU）	仅事务消息	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 税务 ID 或许可证号 • 商业登记证书 • 联系人电子邮件地址 • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 确认此使用案例将适用于从该账户发送到俄罗斯的所有消息 • 确认必须使用单独的发送人 ID 发送非事务消息

国家/地区名称	消息类型	格式限制和要求	注册要求
			• 费用为 272 美元/月，请确认您批准此定期每月费用：是/否 • 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
沙特阿拉伯（SA）	每个发送人 ID 必须要么为事务型，要么为促销型。一个发送人 ID 不能用于这两种类型的流量。如果发送流量OTP或 2FA 流量，则发件人 ID 只能用于此目的。促销发件人IDs将受沙特阿拉伯的 “请勿打扰” (DND) 列表的约束。	• 促销型发送人 ID 长度：2 - 8 个字符，发送人 ID 前面带有“-AD” • 事务型发送人 ID 长度：2 - 11 个字符 • 发送人 ID 必须代表发送人的品牌身份 • 至少包含一个字母 • 请勿使用ASCII特殊字符（例如 #、@） • 可以包括大写和小写字母，以及数字 0 - 9	仅支持跨国公司在沙特阿拉伯注册发件人 ID。我们目前不支持沙特阿拉伯当地公司注册发件人 IDs • 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 您计划发送的消息模板。 • 此发送人 ID 会用于事务或促销内容吗？

国家/地区名称	消息类型	格式限制和要求	注册要求
			• 确认必须使用单独的发送人 ID 发送非事务消息 • 如果发送双重身份验证或OTP流量，请确认此发件人 ID 将ONLY用于此目的
新加坡（SG）	仅事务消息	• 最多 11 个字符 • 没有空格 • 无特殊字符	请参阅 新加坡的 Amazon SNS 发件人 ID 注册要求。

国家/地区名称	消息类型	格式限制和要求	注册要求
斯里兰卡（LK）	没有限制或特殊要求	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司类型（当地/跨国） • 公司URL（链接到您的应用程序或公司网站） • 使用案例说明，消息的用途 • 您计划发送的消息模板 • 此发送人 ID 会用于事务或促销内容吗？ • 确认必须使用单独的发送人 ID 发送非事务消息 • 如果发送双重身份验证或OTP流量，请确认此发件人 ID 将仅用于此目的

国家/地区名称	消息类型	格式限制和要求	注册要求
泰国（TH）	没有限制或特殊要求	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 发送SMS的类型（交易/ OTP 促销/） • 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
土耳其 (TR)	没有限制或特殊要求	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司URL（链接到您的应用程序或公司网站） • 如果您的公司是土耳其当地公司或跨国公司 • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 发送SMS的类型（交易/ OTP 促销/） • 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
乌克兰（UA）	没有限制或特殊要求	• 字母数字 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司URL（链接到您的应用程序或公司网站） • VAT数字 • 当地公司或跨国公司 • 每月估计的发送量 • 使用案例说明，消息的用途 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 发送SMS的类型（交易/ OTP 促销/） • 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
阿拉伯联合酋长国（AE）	仅事务消息	• 字母数字 • 不允许使用笼统的发送人 ID，且必须指明品牌 • 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 公司名称 • 公司地址（包括公司所在城市、州/省、邮政编码） • 公司所在国家/地区 • 公司联系电话号码 • 公司URL（链接到您的应用程序或公司网站） • 每月估计的发送量 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 使用案例说明，消息的用途 • 公司的官方营业/贸易许可证号或 # VAT • 书面确认此发送人 ID 发送的所有流量均涵盖在给出的使用案例描述中 • 您计划发送的消息模板

国家/地区名称	消息类型	格式限制和要求	注册要求
越南（VN）	仅事务消息。不允许营销和促销消息。禁止的内容包括： • 成人内容 • 慈善服务 • Cryptocurrency • 乐透 • 手机赌博/赌场 • 体育博彩 • 投票	• 最多 11 个字符 • 没有空格 • 无特殊字符	• 要注册的发送人 ID • 用户将从哪个 AWS 区域调用 API / service • 每月估计的发送量 • 如果不明显，请解释公司名称和发送人 ID 之间的联系 • 使用案例说明，消息的用途 • 书面确认此发送人 ID 发送的所有流量均涵盖在给出的使用案例描述中

签名格式限制和要求

要成功发送到中国，您需要随消息内容模板注册消息签名。每次SMS发送的消息内容都必须加上此签名。不在SMS邮件正文前面加上注册的签名可能会导致被屏蔽或过滤SMS。

- 正文前面必须加上方括号SMS内的签名
- 标准文本必须包含在方括号内
- Unicode 文本必须使用柱状方括号来包含签名 — U+3010 LEFT LENTICULAR BRACKET 和 U RIGHT LENTICULAR BRACKET +3011 — 示例：[注意]
- 长度必须介于 3 – 11 个字符之间。
- 支持中文/英文字符

法国的亚马逊SNS发件人编号要求

本指南提供了必要的步骤和指南，以创建法国移动运营商向法国发送SMS短信所需的专用发件人 ID。

主题

- 为法国设置专用发件人 ID

- 发件人 ID 命名准则

为法国设置专用发件人 ID

您可以使用以下方法之一设置专用发件人 ID。对于使用发布的SMS邮件，Amazon SNS 将代表您使用发件人 ID Publish API。

- 您可以使用 Amazon SNS 控制台配置用于所有已发布SMS消息的默认发件人 ID。要了解更多信息，请访问使用设置SMS消息首选项 AWS Management Console。
- 在PublishAPI请求 Amazon SNS 发布消息时，您可以使用AWS.SNS.SMS.SenderID消息属性设置发件人 ID。SMS要了解更多信息，请访问使用控制台发送消息。

发件人 ID 命名准则

- 发件人 ID 名称必须为字母数字，最多为 11 个字符。
- 发件人 ID 名称不得包含特殊字符或空格。
- 我们建议您使用相同的名称作为发件人 ID 和发送SMS短信的公司的品牌名称。

印度的 Amazon SNS 发件人 ID 注册要求

默认情况下，当您向印度的收件人发送消息时，Amazon SNS 使用国际长途运营商 (ILDO) 连接来传输这些消息。当收件人看到通过ILDO连接发送的消息时，该消息似乎是通过随机数字 ID 发送的（除非您购买了专用的短代码）。

> **Note**
>
> 使用本地路由发送消息的价格显示在 Amazon SNS 全球SMS定价页面上。使用ILDO连接发送消息的价格高于通过本地路由发送消息的价格。

如果您更喜欢使用按字母顺序排列的发件人 ID 发送SMS邮件，则必须通过本地路由而不是ILDO路由发送这些邮件。要使用本地路由发送消息，必须先通过分布式账本技术 () 门户向印度电信监管局 (TRAI) 注册您的用例和消息模板。DLT这些注册要求旨在减少印度使用者收到的未经请求的消息数量，并保护使用者免受潜在有害消息的影响。此注册过程是由 Vodafone （印度）通过其 Vilpower 服务管理的。

主题

第 1 步：向注册 TRAI

在向印度的收件人发送SMS消息之前，您必须向印度电信监管局（TRAI）注册您的组织。在注册过程中，准备好提供以下信息：

- 贵组织的永久账号 (PAN)。
- 贵组织的税收减免账号 (TAN)。
- 贵组织的商品和服务税识别号 (GSTIN)。
- 贵组织的企业标识号 (CIN)。
- 授权您注册您的组织的授权书。

以下是一些分布式账本技术 (DLT) 注册网站的示例列表，您可以使用这些网站注册您的组织TRAI（可能需要付费）。注册过程因站点而异。请联系他们各自的支持团队以寻求帮助。

- BSNLDLT-免费注册。
- Jio TrueConnect - 收取完成注册流程的费用。
- Smart Enterprise Solutions - 收取完成注册流程的费用。
- Vilpower - 包含一个模板，您可以根据需要下载并修改此模板。Vilpower 对完成注册过程会收取一定费用。

要将您的组织注册到 TRAI

以下详细介绍了如何TRAI使用 Vilpower 向您的组织注册。

1. 在 Web 浏览器中，转到 Vilpower 网站，网址为 https://www.vilpower.in。
2. 选择 Signup（注册）以创建另一个账户。在注册过程中，请执行以下操作：
   - 对于要注册为的实体类型，请选择 As Enterprise（作为企业）。
   - 对于电话推销员姓名，请使用 Infobip P rivate Limited-。ALL出现提示时，开始键入，**`Infobip`**然后ALL从下拉列表中选择 Infobip Private Limit ed。

- 在输入电话营销人员 ID中，输入 **110200001152**。
- 当系统提示您提供标题时IDs，请输入您IDs要注册的发件人。

> **Note**
>
> 印度要求发件人的长度IDs必须恰好为六个字符。

- 当系统提示您提供内容模板时，请输入您计划发送给接收人的消息内容。为您计划发送的每条消息包含一个模板。

> **Note**
>
> DLT注册提供商的网站不由 Amazon Web Services 维护。网站上的步骤可能会随时更改。

步骤 2：请求发件人 ID

要在印度申请发件人ID，您需要提交 AWS Support 申请。完成 正在请求发件人 IDs 中的步骤。在您的请求中，提供以下必需信息：

- 发件人计划从中发送SMS消息的 AWS 区域。
- DLT注册过程中使用的公司名称。
- 成功注册实体后收到的委托人DLT实体 ID (PEID)。
- 每月估计的交易量。
- 您的使用案例的解释。
- 最终用户选择加入流的说明。
- 确认收集并注册最终用户选择项。

步骤 3：发送SMS消息

在注册组织后TRAI，您可以向印度的收件人发送SMS消息。

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，将区域选择器设置为支持消息SMS传递的区域。
3. 在导航面板上，选择短信 (SMS)。

4. 在移动短信 (SMS) 页面上，选择发布短信。将打开 “发布SMS消息” 窗口。
5. 对于 Message type，请选择下列选项之一：

   - 促销 – 不重要的消息，例如营销消息。

     使用数字发送器时IDs，请选择此选项。

   - Transactional（事务性）– 为客户事务处理提供支持的重要消息，例如多重身份验证的一次性密码。

     使用字母或字母数字发件人时IDs，请选择此选项。

   此消息级别的设置会覆盖您在 Text messaging preferences 页面设置的默认消息类型。

   有关促销和交易消息的定价信息，请参阅全球SMS定价。
6. 对于数字，请输入您想要向其发送消息的电话号码。
7. 对于 Message（消息），请输入要发送的消息。

   向SMS消息添加内容时，请确保其与DLT注册模板中的内容完全匹配。如果运营商的SMS消息内容包含额外的返回字符、空格、标点符号或不匹配的句子大小写，则会屏蔽消息。模板中的变量可以包含 30 个或更少的字符。
8. 在发起身份部分，对于发件人 ID，输入包含 3-11 个字符的自定义 ID。

   对于促销消息，发件人IDs可以是数字，对于交易消息，发件人可以是字母或字母数字。该发件人 ID 在接收设备上显示为消息发件人。

   对于在印度IDs注册的数字促销发件人，请在SMS发送请求中将发件人 ID 指定为发件人编号参数。
9. 展开 “特定国家/地区属性” 部分，指定以下向印度收件人发送SMS邮件所需的属性：

   - 实体 ID — 您从监管机构收到的用于向印度收件人发送SMS消息的实体 ID 或委托人实体 (PE) ID。

     这是一个自定义的、TRAI由 1—50 个字符组成的字符串，用于唯一标识您在中注册的实体。TRAI
   - 模板 ID — 您从监管机构收到的用于向印度收件人发送SMS消息的模板 ID。

     这是一个自定义的、TRAI由 1—50 个字符组成的字符串，用于唯一标识您在中注册的模板。TRAI模板 ID 必须与您在上一步中指定的发件人 ID 以及消息内容相关联。

10. 选择发布消息。

有关向其他国家/地区的收件人发送SMS消息的信息，请参阅使用 Amazon 向手机发布SMS消息SNS。

对SMS发送给印度收件人的邮件进行故障排除

以下是运营商可能屏蔽SMS消息的一些原因：

- No template was found that matched the content sent.（未找到与发送内容匹配的模板。）

  发送的内容：**<#> 12345 is your OTP to verify mobile number. Your OTP is valid for 15 minutes -- ABC Pvt. Ltd.**

  匹配的模板：无

  问题：DLT注册DLT模板的开头没有包含<#>或{#var#}的模板。

- The value of a variable exceeds 30 characters.（变量的值超过 30 个字符。）

  发送的内容：**12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  匹配的模板：**{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC Pvt. Ltd.**

  问题：发送的内容中 “ABC公司-印度私人有限公司” 的值超过了30个{#var#}字符的限制。

- The message sentence case does not match the sentence case in the template.（消息句子大小写与模板中的句子大小写不匹配。）

  发送的内容：**12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  匹配的模板：**{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC PVT. LTD.**

  问题：附加到DLT匹配模板的公司名称是大写的，而发送的内容已将部分名称更改为小写——“ABCPvt.Ltd.” vs. “ABCPVT。LTD。”

新加坡的 Amazon SNS 发件人 ID 注册要求

Amazon SNS 客户可以使用已通过新加坡发件人 ID 注册中心 (SSIR) 注册的SMS发件人 ID 在新加坡发送SMS流量。SSIR于 2022 年 3 月通过新加坡信息通信媒体发展局 (SGNIC) 旗下的新加坡网络信息中心 (IMDA) 推出，允许组织在向新加坡的手机发送SMS数据时注册其发件人ID。

要使用已注册的新加坡发件人ID，您必须获得一个唯一的实体编号 (UEN)，然后向亚马逊提交申请，将您的账户列入允许使用您的发件人编号的名单，最后通过完成注册流程。SSIR

如果您未在 2023-01-30 之前注册身份证，则根据监管机构的规定，任何使用发件人 ID 发送的邮件的 ID 都将更改为LIKELY。SCAM在此日期之后，监管机构将继续自行筛选或阻止未注册的流量。

Important

如果您在 Amazon Pinpoint 区域申请了发送人 ID，请使用 Amazon Pinpoint 控制台注册发送人 ID。要手动完成 Amazon Pinpoint 区域以外区域的注册流程，请使用新加坡发送人 ID 注册。
为确保您仍然可以在新加坡发送消息，您的注册必须在 2023 年 1 月 30 日之前完成。
按以下顺序完成注册步骤非常重要。如果未按顺序执行这些步骤，则可能导致您的发件人 ID 受到服务阻止，或者导致您的发件人 ID 无法保留在移动设备上。
步骤 1.注册新加坡唯一实体编号 (UEN)
第 2 步。如果您在 Amazon Pinpoint 区域申请了发送人 ID，请按照 Amazon Pinpoint 发送人 ID 注册说明来注册发送人 ID。

- 要在账户不在 Amazon Pinpoint 区域的情况下注册发送人 ID，请使用新加坡发送人 ID 注册说明手动注册发送人 ID。
- 代表另一家公司发送SMS短信时，需要该公司的授权书 (LOA)。
- 提交 AWS 发件人 ID 注册后，不要等待批准或状态变更。立即转到步骤 3。

步骤 3.在新加坡网络信息中心注册发件人ID (SGNIC)

主题

- 新加坡发件人 ID 注册状态
- 编辑新加坡发件人 ID 注册
- 删除新加坡发件人 ID 注册
- 新加坡注册问题
- 新加坡发件人 ID 注册常见问题

注册新加坡唯一实体编号 (UEN)

要开始注册，SSIR您必须先获得新加坡唯一实体编号 (UEN)。A UEN 是您在账户和公司注册管理局 (ACRA) 注册业务时收到的唯一实体编号，有关更多信息，请参阅谁必须向谁注册ACRA？ 。处理时间可能会有所不同，具体取决于验证您的请求的ACRA难易程度。

向 Amazon Pinpoint 注册您的新加坡发件人 ID

注册新加坡唯一实体号码 (UEN) 后，即可在亚马逊 Pinpoint 控制台中完成发件人身份注册流程（仅适用于亚马逊 Pinp oint 区域）。注册发件人 ID 时，请确保信息完整准确，否则您的注册可能会被拒绝。

> **Important**
>
> 您通过 Amazon Pinpoint 控制台提交的信息将传递给我们的运营商合作伙伴以完成注册。

注册新加坡发件人 ID

当账户位于 Amazon Pinpoint 区域时，使用这些步骤注册发送人 ID。如果您的账户不在 Amazon Pinpoint 区域，请参阅完成新加坡发件人 ID 注册的手动注册流程。 。

1. 登录 AWS 管理控制台并打开位于的 Amazon Pinpoint 控制台。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的 “SMS和语音” 下，选择 “电话号码”。
3. 在发件人 ID 注册选项卡上，选择创建注册。
4. 选择新加坡作为您的目的地国家/地区。
5. 在 Company Information（公司信息）部分，输入以下内容：
   - 在 “公司名称” 中，输入与UEN注册时完全相同的公司名称。
   - 对于税号，请输入您从中收到的UEN号码ACRA。

- 对于公司网站，请URL输入贵公司网站的。
- 对于 Address 1（地址 1），请输入您的公司总部的街道地址。
- 对于 Address 2 - optional（地址 2 - 可选），如果需要，请输入您的公司总部的房间号。
- 对于 City（城市），请输入您的公司总部所在的城市。
- 对于 State（州），请输入您的公司总部所在的州。
- 对于 Zip Code（邮政编码），请输入您的公司总部的邮政编码。
- 在国家/地区中，输入两位数ISO的国家/地区代码。

6. 在联系信息部分，输入以下信息：
   - 对于 First Name（名字），输入将担任贵公司联系人的人员的名字。
   - 对于 Last Name（姓氏），输入将担任贵公司联系人的人员的姓氏。
   - 对于 Support Email（支持电子邮件），输入将担任贵公司联系人的人员的电子邮件地址。
   - 对于 Support Phone Number（支持电话号码），输入将成为贵公司联系人的人员的电话号码。
7. 在发件人 ID 信息中，输入以下信息：
   - 对于 Sender ID（发件人 ID），请输入要在消息中显示的发件人 ID。
   - 对于是否代表其他品牌/实体注册？，如果是，请选择“是”。如果您不是发送消息的最终用户，则被视为其他品牌/实体的“代表”。
   - 对于授权书图片 — 可选，如果您选中了 “代表其他品牌/实体注册？” 复选框？ ，上传完整授权书的图片 (LOA)。支持的文件类型为PNG，最大文件大小为 400KB。为方便起见，LOA可以下载的模板。
   - 对于 Sender ID connection – optional（发件人 ID 关联 – 可选），您可以添加有关申请的发件人 ID 与公司名称之间关联的详细信息。
8. 在 Messaging Use Case（消息收发使用案例）中，执行以下操作：
   - 在 “每月SMS数量” 中，选择每月的SMS消息数量。
   - 对于 Use Case Category（使用案例类别），为该号码选择以下使用案例类型之一：
     - Two-factor authentication（双重身份验证）– 用于发送双重身份验证码。
     - One-time passwords（一次性密码）– 用于向用户发送一次性密码。
     - Notifications（通知）– 如果您只想向用户发送重要通知，请使用此选项。
     - Polling and surveys（轮询和调查）– 用于轮询用户的偏好。
     - Info on demand（按需消息）– 用于在用户发送请求后向他们发送消息。
     - Promotions and Marketing（促销和市场营销）– 如果您只想向用户发送市场营销信息，请使用此选项。

- Other（其他）– 如果您的使用案例不属于任何其他类别，请使用此选项。请务必填写此选项的 Use Case Details（使用案例详细信息）。
- 完成 Use Case Details（使用案例详细信息）– 可选，用于为所选 Use Case Category（使用案例类别）提供更多上下文。

9. 在 Messaging Samples（消息收发示例）部分中，执行以下操作：
   - 在消息示例 1 中，输入将发送给最终用户的SMS消息正文示例。
   - 对于消息示例 2（可选）和消息示例 3（可选），输入要发送的SMS消息正文的其他示例消息（如果需要）。
   - 每个 Message Sample（消息示例）文本框的最大字符限制为 306 个字符。
10. 完成此操作后，选择 Submit registration（提交注册）。

> **Important**
>
> 您可以按照新加坡发件人 ID 注册状态中的说明检查您的注册状态。
> 提交发送人 ID 注册后，不必等待批准或状态变更。立即转至在新加坡网络信息中心注册发件人ID (SGNIC)。

完成新加坡发件人 ID 注册的手动注册流程。

当账户不在 Amazon Pinpoint 区域时，使用以下步骤注册发送人 ID。如果您的账户在 Amazon Pinpoint 区域中，请参阅向 Amazon Pinpoint 注册您的新加坡发件人 ID。

1. 下载 Singapor e_sender_id_Registration_ _Template.zip 并填写所需信息LOA。
2. 向 AWS Support 创建案例。
3. 在 Open support cases（提交支持案例）选项卡上，选择 Create case（创建案例）。
4. 选择 “查找服务限制增加”，对于限制类型，选择 “SNS短信”。
5. 对于资源类型，选择发件人 ID 注册。
6. 附上LOA文件并提交申请。

在新加坡网络信息中心注册发件人ID (SGNIC)

> **Warning**
>
> 如果未按顺序执行这些步骤，则可能导致您的发件人 ID 受到服务阻止，或者导致您的发件人 ID 无法保留在移动设备上。

1. 您必须先在 AWS 为您的账户注册新加坡（SG）发件人 ID（通过 Amazon Pinpoint 控制台注册，对于非 Amazon Pinpoint 区域，则手动注册）。完成此步骤后，可继续执行下一步。
2. 使用发件人 ID 注册表中的流程注册您的SGNICSMS发件人 ID。SGNIC
   - 完成该过程后，请确保将以下所有内容列为参与聚合者：
     - AMCSSG Private Limited（亚马逊媒体通信服务）
     - Nexmo PTE LTD
     - 新加坡辛奇 PTE LTD
     - 新加坡电信 PTE LTD
     - Twilio 新加坡 PTD LTD

> **Note**
>
> 您需要从每个 AWS 账户中提交发件人 ID 注册才能使用发件人 ID。

新加坡发件人 ID 注册状态

当您在 Amazon 上注册新加坡发件人编号时，SNS您的注册将处于以下五种不同的状态之一：

- Created（已创建）– 您的注册已创建但尚未提交。
- 已提交 – 您的注册已提交，正在接受验证。
- Reviewing（正在审核）– 您的注册已被接受，正在接受审核。审核过程可能需要 1-3 周，在某些情况下可能需要更长时间。
- 完成 – 您的注册已获批准，您可以开始使用发件人 ID。
- Requires Updates（需要更新）– 您需要修正注册信息并重新提交。请参阅编辑新加坡发件人 ID 注册了解更多信息。需要更新的字段将会显示一个警告图标和问题的简要描述。

对于除 Amazon Pinpoint 区域以外的所有区域，AWS Support 将在注册时发送电子邮件确认，或向 AWS Support提交案例。

- 在 Open support cases（提交支持案例）选项卡上，选择 Create case（创建案例）。
- 选择提高服务限制。
- 对于资源类型，请选择 Sender ID Registration（发件人 ID 注册），对于限制，请选择 General Inquiry（一般查询）。

检查注册状态

1. 登录 AWS 管理控制台并打开位于的 Amazon Pinpoint 控制台。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的 “SMS和语音” 下，选择 “电话号码”。
3. 在发件人 ID 注册选项卡上，选择发件人 ID。
4. 您随后可以查看每个发件人 ID 的注册状态。

编辑新加坡发件人 ID 注册

提交在 Amazon Pinpoint 的注册后，如果注册存在问题，则注册状态将设置为需要更新。在此状态下，注册表单可供编辑。需要更新的字段将显示一个警告图标和问题的简要描述。

编辑发件人 ID

1. 打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的 “SMS和语音” 下，选择 “电话号码”。
3. 在发件人 ID 注册选项卡上，选择要编辑的号码并选择注册 ID。
4. 选择 Update registration（更新注册）以编辑表单并更正带有警告图标的字段。
5. 如果您代表其他品牌/实体注册，则需要为授权书图像 – 可选重新上传之前提交的文件。
6.

> **Important**
>
> 重新检查所有字段以确保正确无误。

7. 完成此操作后，选择 Submit registration（提交注册）以重新提交。

删除新加坡发件人 ID 注册

如果您不想继续进行新加坡发件人 ID 注册，则可以删除注册。只有状态为已创建或需要更新的注册才能删除。

删除注册

1. 打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的 “SMS和语音” 下，选择 “电话号码”。
3. 在发件人 ID 选项卡上，选择要删除的注册 ID 并选择删除 ID。

新加坡注册问题

如果您的新加坡发件人 ID 未被 Amazon Pinpoint 接受，您将看到一条消息，说明拒绝它的原因。如果我们的最佳实践中未解答您对此拒绝的疑问，您可以向我们的支持团队提交请求。

提交有关被拒绝的新加坡发件人 ID 的信息请求

1. 打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 选择支持，然后选择支持中心。
3. 在“支持”页面上，选择创建案例。
4. 对于 Case type（案例类型），请选择 Service limit increase（提高服务限制）。
5. 对于 Limit type（限制类型），选择 PinpointSMS。
6. 在 Requests（请求）部分中，执行以下操作：
   - 对于 Resource Type（资源类型），请选择 Sender ID Registration（发件人 ID 注册）。
   - 对于 Limit（限制），选择 Registration Rejection Query（拒绝注册查询）。
7. 在 Use case description（使用案例描述）中，输入被拒绝的新加坡发件人 ID 和提供的拒绝原因。
8. 在 “联系人选项” 下，在 “首选联系语言” 中，选择您与 Su AWS pport 团队沟通时首选使用的语言。
9. 对于联系方式，请选择您首选的与 Su AWS pport 团队沟通的方法。
10选择提交。

在您的 AWS 支持案例中，支持团队将提供有关您的发件人 ID 注册被拒绝的原因的信息。 AWS

新加坡发件人 ID 注册常见问题

有关通过 Amazon Pinpoint 注册新加坡发件人 ID 号码的常见问题。

我现在有新加坡发件人 ID 吗？

检查您是否拥有新加坡发件人 ID

1. 打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的 “SMS和语音” 下，选择 “电话号码”。
3. 在发件人 ID 注册选项卡上，选择您要查看的发件人 ID，然后选择注册 ID。

注册需要多长时间？

虽然一般的审查需要 1 到 3 周，但在某些情况下，向政府机构验证您的信息可能需要长达 5 周或更长时间。

什么是唯一实体编号 (UEN)？如何获得？

A UEN 是由会计和公司监管局 (ACRA) 签发的新加坡商业身份证。新加坡的本地公司和企业UEN可以通过申请获得ACRA。在您通过了登记和标准公司注册程序后就会签发。你可以ACRA通过 Bizfile 申请with。UEN

我必须注册新加坡发件人 ID 吗？

是。如果您目前尚未在 2023-01-30 之前注册您的新加坡发件人 ID，则任何使用发件人 ID 发送的邮件的 ID 都将更改为-LIKELY SCAM

如何通过 Amazon Pinpoint 注册我的新加坡发件人 ID？

请按照“向 Amazon Pinpoint 注册您的新加坡发件人 ID”中的说明注册发件人 ID。

我的新加坡发件人 ID 处于什么注册状态？这是什么意思？

按照“新加坡发件人 ID 注册状态”中的说明检查您的注册和状态。

我需要提供哪些信息？

您需要提供您的公司地址、业务联系人和使用案例。您可以在“向 Amazon Pinpoint 注册您的新加坡发件人 ID”中找到所需信息。

如果我的新加坡发件人 ID 注册遭到拒绝，该怎么办？

如果您的注册遭到拒绝，其状态将更改为“Requires Updates”（需要更新），您可以按照“编辑新加坡发件人 ID 注册”中的说明进行更新。

我需要哪些权限？

您用于访问 Amazon Pinpoint 控制台的IAM用户/角色必须使用 *"sms-voice:*"* 许可。

了解 Amazon 中的原产地编号 SNS

发件人号码是一个数字字符串，用于标识SMS消息发件人的电话号码。当您使用发件人号码发送SMS消息时，收件人的设备会将发件人号码显示为发件人的电话号码。您可以按使用案例指定不同的源号码。

> **Tip**
>
> 要查看您 AWS 账户中所有现有发货编号的列表，请在亚马逊SNS控制台的导航窗格中选择 Originat ion 编号。

在当地法律要求使用发件人IDs代替发件人号码的国家/地区，不提供对发件人号码的支持。

主题

- 亚马逊 SNS 10 DLC
- Amazon SNS 免费电话号码
- Amazon SNS 专用短码
- 亚马逊SNS person-to-person 长码
- Amazon SNS 美国商品编号对比

亚马逊 SNS 10 DLC

美国运营商不再支持通过本地、未注册的长码使用 application-to-person (A2P) SMS 消息传递。对于大容量的 A2P SMS 消息，美国运营商改为提供一种名为 10 位长码 (10) 的新型长码。DLC

> **Important**
>
> 从 2023 年 1 月 26 日起，亚马逊SNS的SMS供应商对 10 个DLC活动引入了新的手动审查流程，以解决美国承运人提出的SMS垃圾邮件问题。您可以使用短代码和免费电话号码作为10的替代方案DLC，SMS在美国发送。
> 目前，我们的SMS供应商尚未提供服务级别目标，说明10次DLC活动审查需要多长时间。一旦数字与 10 DLC 广告活动相关联，就会触发评论。评论所花费的时间超过SNS了亚马逊先前公布的 14 天预计时间。

Amazon SNS 每天都在与SMS供应商合作，以确保：

- 供应商尽快完成所有待处理的 10 条DLC活动审核
- 供应商在待办事 AWS 项中对请求进行优先排序

您可以按照 10 个DLC广告系列的说明查看 10 个DLC广告系列的状态。如果需要其他信息才能批准 10 DLC 活动， AWS 支持团队将通知您。
注册美国免费电话号码的速度可能比获得 10 个DLC号码更快。有关美国免费电话号码和注册流程的更多信息，请参阅免费电话号码注册要求和流程。

什么是 10DLC？

10 DLC 是一种向运营商注册的长代码，用于支持使用 10 位数电话号码格式的大量 A2P SMS 消息。亚马逊SNS不再将本地长码作为SMS产品提供，而是提供 10 个DLC。如果您只使用短代码和免费电话号码，则10 DLC 不会对您产生影响。

10 DLC 是一个 10 位数的电话号码，仅在美国使用。从 10 发送DLC给收件人的邮件显示一个 10 位数的数字作为发件人。与免费电话号码不同，10 DLC 支持交易和促销消息，并且可以包含任何美国区号。

如果您已有本地长码，则可以请求为 10 启用本地长码DLC。为此，请完成 10 DLC 注册流程，然后提交支持请求。如果在为 10 启用长代码时出现问题DLC，系统会通知您并指示您DLC通过 Amazon Pinpoint（不是亚马逊SNS）控制台请求新的 10。有关如何提交支持票证以转换长代码的信息，请参阅 在 Amazon 中将长代码与 10 DLC 广告系列关联 SNS。

要使用 10 DLC 数字，请先注册您的公司，然后使用亚马逊 Pinpoint（不是亚马逊SNS）控制台创建 10 DLC 的活动。 AWS 与 Campaign Registry 共享此信息，后者是根据这些信息批准或拒绝您的注册的第三方。在某些情况下，立即开始注册。例如，如果您之前已在“活动注册处”注册，则它们可能已经拥有您的信息。但是，有些活动可能需要一周或更长时间才能获得批准。在您的公司和 10 个DLC广告活动获得批准后，您可以购买 10 个DLC号码并将其与您的广告活动关联。申请 10 也DLC可能需要长达一周的时间才能获得批准。尽管您可以将多个 10 DLCs 与单个广告活动关联，但不能在DLC多个广告系列中使用相同的 10。对于你创建的每个广告系列，你都需要有一个唯一的 10 DLC。

10 种DLC能力

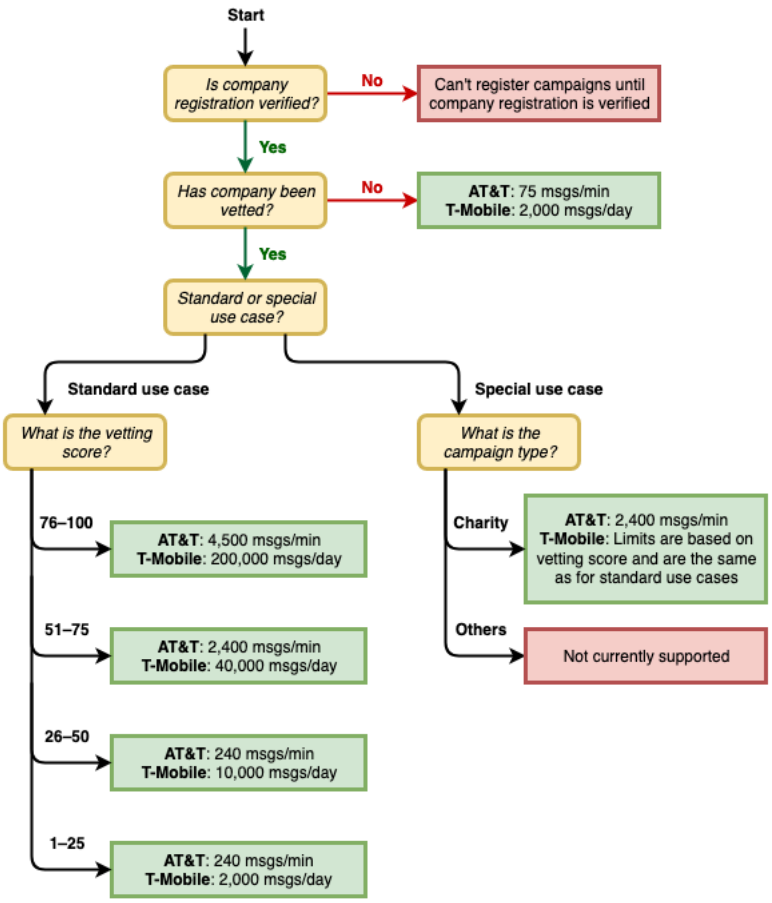
10 个DLC电话号码的功能取决于您的收件人使用的移动运营商。AT&T 对每个活动每分钟可以发送的消息部分数量设有限制。T-Mobile 对每家公司每日可发送消息设有限制，对每分钟可发送的消息部分

数量没有限制。Verizon尚未发布吞吐量限制，但使用了10的过滤系统DLC，该系统旨在删除垃圾邮件、未经请求的邮件和滥用内容，而不太重视实际的邮件吞吐量。

与被审查的公司相关的新10个DLC活动可以每分钟向使用AT＆T的收件人发送75条消息，每天向使用T-Mobile的收件人发送2,000条消息。公司限额在您的所有 10 个DLC广告系列中共享。例如，如果您注册了一家公司和两个活动，则每天向 T-Mobile 客户发送 2000 条消息的限制将应用于这些活动。同样，如果您在多个 AWS 账户中注册同一家公司，则每日配额将在这些账户之间共享。

如果您的吞吐量需求超出这些限制，可以请求审查您的公司注册。当您审查公司注册时，第三方验证提供商会分析您的公司详细信息。然后，验证提供商会提供审查分数，该分数决定了你的 10 个DLC广告活动的能力。审查服务收取一次性费用。有关更多信息，请参阅 正在审核您的 Amazon SNS 10 注册信息 DLC。

您的实际吞吐率将取决于各种因素，例如贵公司是否已经过审查、活动类型以及审查分数。以下流程图显示了各种情况的吞吐率。



10 DLC 的吞吐率由美国移动运营商与活动注册局合作确定。Amazon 和任何其他SMS发送服务都SNS无法在这些速率之外增加 10% 的DLC吞吐量。如果您需要在所有美国运营商中实现高吞吐速率和高送

达率，建议您使用短代码。有关获取短代码的更多信息，请参阅向 Amazon 申请SMS发送消息的专用短代码 SNS。

10 入门 DLC

使用亚马逊 Pinpoint 控制台（不是亚马逊SNS）申请你的 10。DLC按照以下步骤设置 10，DLC用于你的 10 个DLC广告系列。

1. 注册您的公司。

   在申请 10 之前DLC，您的公司必须在 “活动注册表” 中注册；有关信息，请参阅亚马逊 SNS 10 DLC 公司注册流程。注册通常是即时的，除非活动注册处需要更多信息。注册您的公司需要支付一次性注册费，该费用显示在注册页面上。这笔一次性费用与您每月的活动费用分开支付，10DLC.

   > **Note**
   >
   > 亚马逊SNSSMS消息在目前不支持 Amazon Pinpoint 的地区可用。有两种不同的情况：
   >
   > a. 如果您使用的是商用云账户，则需要在美国东部（弗吉尼亚北部）地区打开 Amazon Pinpoint 控制台来注册您的 10 DLC 家公司和活动。不要请求 10 DLC 号码。
   >
   > b. 使用 S AWS ervice Q uotas 控制台创建服务限制提高案例，同时请求该区域的 10 DLC 数字。有关已推出 Amazon Pinpoint 的区域的信息，请参阅《AWS 一般参考》中的 Amazon Pinpoint 端点和限额。
   >
   > c. 如果您使用的是AWS GovCloud (US)账户，请打开美国西部地区的 Amazon Pinpoint 控制台，注册您的 10 DLC 家公司和活动。不要请求 10 DLC 号码。相反，请使用 S AWS ervice Q uotas 控制台创建服务限制提高案例，同时请求该区域的 10 DLC 数字。有关已推出 Amazon Pinpoint 的区域的信息，请参阅《AWS 一般参考》中的 Amazon Pinpoint 端点和限额。

2. （可选，但推荐）申请审查

   如果您的公司注册成功，则可以开始制作小批量、混合用途 10 个广告系列。DLC这些活动每分钟可以向使用 AT&T 的接收人发送 75 条消息，您注册的公司每天可以向使用 T-Mobile 的接收人发送 2000 条消息。如果您的使用案例要求的吞吐率超过这些值，可以申请对公司注册进行审查。审查您的公司注册可以提高公司和活动的吞吐率，但不能保证。有关审查的更多信息，请参阅正在审核您的 Amazon SNS 10 注册信息 DLC。

3. 注册您的活动。

在您的公司注册后，创建一个 10 个DLC广告系列并将其与您的注册公司关联起来。此活动将提交给活动注册处以供批准。在大多数情况下，除非DLC竞选登记处要求提供更多信息，否则 10 个活动即时获得批准。有关更多信息，请参阅 亚马逊 SNS 10 DLC 活动注册。

4. 申请您的 10 DLC 号码。

   在您的 10 个DLC广告活动获得批准后，您可以申请 10 DLC 并将该数字与已批准的活动关联起来。你的 10 DLC 广告活动只能使用批准的数字。请参阅 请求 10 个DLC号码、免费电话号码和 P2P 长码，以便SMS通过 Amazon 发送消息 SNS。

10 DLC 注册费和月费

使用 10（例如注册您的公司和 10 个DLCDLC广告活动）需要支付注册费和月费。这些费用与 AWS收取的任何其他月度费用是分开的。有关更多信息，请参阅 Amazon SNS 全球SMS定价页面。

亚马逊 SNS 10 DLC 公司注册流程

在申请 10 之前DLC，您需要在竞选注册处注册您的公司。

> **Note**
>
> 亚马逊SNSSMS消息在目前不支持 Amazon Pinpoint 的地区可用。在这种情况下，请打开美国东部（弗吉尼亚北部）地区的 Amazon Pinpoint 控制台注册您的 10 DLC 家公司和活动，但不要申请 10 DLC 号码。相反，请使用 S AWS ervice Quotas 控制台创建服务限制提高案例，同时请求该区域的 10 DLC 数字。有关已推出 Amazon Pinpoint 的区域的信息，请参阅《AWS一般参考》中的 Amazon Pinpoint 端点和限额。

10 个DLC公司注册状态

注册公司或品牌时，将返回以下两种状态之一：未验证或已验证。如果您的公司注册状态为未验证，表明您的注册存在问题。例如，您提供的注册公司名称可能与您提供的税号相关联的公司的注册名称不完全匹配。如果您发现公司注册详细信息有问题，可以更正。有关修改公司注册详细信息的更多信息，请参阅编辑或删除 Amazon SNS 10 DLC 注册公司。

如果您的公司注册状态为 “已验证”，则您提供的注册详细信息是准确的，您可以开始创建 10 个DLC广告系列。

注册您的公司或品牌

您只需对您的公司进行一次注册。注册后，您可以编辑公司和联系信息。要删除注册的公司，请使用 AWS Support 创建案例。有关编辑或删除公司详细信息的更多信息，请参阅编辑或删除 Amazon SNS 10 DLC 注册公司。

注册公司

1. 登录 AWS Management Console 并打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的 “SMS和语音” 下，选择 “电话号码”。
3. 在 10 个DLC广告系列选项卡上，选择注册公司。

> **Note**
>
> 注册您的公司页面会显示注册费。该费用是注册您的公司所产生的相关一次性费用。此费用与任何其他月度成本或费用分开。当您注册公司或修改现有公司注册的详细信息时，将向您收取费用。

4. 在公司信息部分，执行以下操作：

   - 对于合法的公司名称，输入公司注册时所使用的名称。您输入的名称必须与您提供的税号关联的公司名称完全匹配。

> **Important**
>
> 确保使用您公司的确切法定名称。提交后，您将无法更改此信息。不正确或不完整的信息可能会导致您的注册被延误或被拒绝。

   - 对于该组织的法律形式是什么，选择最贴切地描述您的公司的选项。

> **Note**
>
> 美国政府和 N ot-for-profit 选项只能用于注册总部设在美国的组织。如果您的组织位于美国以外的其他国家/地区，必须注册为私人营利组织，无论组织的实际法律形式如何，都是如此。

- 如果在上一步中选择了上市营利组织，则输入公司的股票代号和上市的证券交易所。
- 对于注册国家/地区，选择注册公司所在的国家/地区。
- 对于 Do iness As (DBA) 或品牌名称，请输入贵公司开展业务的任何其他名称。
- 对于 Tax ID（税务 ID），输入您的公司的税务 ID。您输入的 ID 取决于注册您的公司所在的国家/地区。
  - 如果您要注册具有IRS雇主识别码 (EIN) 的美国或非美国实体，请输入您的九位EIN数字。您输入的法定公司名称和实际地址必须与注册的公司信息一致IRS。EIN
  - 如果您注册的是加拿大实体，请输入您的联邦或省级公司号码。请勿输入公司提供的公司编号 (BN) CRA。您输入的法定公司名称、公司号码和实际地址必须与在加拿大公司局注册的公司信息一致。
  - 如果您注册的实体位于另一个国家/地区，请输入您所在国家/地区的主要税号。在许多国家/地区，这是您的VAT身份证号码的数字部分。
- 对于垂直行业，选择最能描述您注册的公司的类别。

5. 在联系信息部分，执行以下操作：

- 对于地址/街道，输入与您的公司关联的实际街道地址。
- 对于城市，输入实际地址所在的城市。
- 对于州或地区，输入地址所在的州或地区。
- 在邮政编码/邮政编码中，输入地址的ZIP或邮政编码。
- 对于公司网站，请输入贵公司网站URL的完整内容。网站地址须以“http://”或“https://”开头。
- 对于支持电子邮件，输入电子邮件地址。
- 对于支持电话号码，输入带有国家/地区代码的电话号码。

> **Note**
>
> Campaign Registry 要求提供联系电子邮件地址和电话号码，因为他们可能需要向您的公司代表验证注册信息。

6. 完成后，选择 Create（创建）。您的公司注册被提交到 Campaign Registry。大多数情况下，该组织会立即受理您的注册并提供一个状态。

如果您的公司注册状态为 “已验证”，则可以开始创建小批量、混合用途 10 个广告系列。DLC您可以使用此类活动每分钟向使用 AT&T 的接收人发送 75 条消息，您注册的公司每天可以向使用 T-Mobile 的接收人发送 2000 条消息。您还可以向使用其他美国运营商（例如 Verizon 和 US Cellular）的接收人发送消息。这些运营商并不严格执行吞吐量限制，但他们会严格监控 10 DLC 封邮件是否有垃圾邮件和滥用行为的迹象。

如果您的使用案例要求的吞吐率超过这些值，可以申请对公司注册进行额外审查。有关审查您的品牌注册的更多信息，请参阅正在审核您的 Amazon SNS 10 注册信息 DLC。

如果公司注册状态为 Unverified（未验证），说明您提供的信息有问题。请检查您提供的信息并确认所有字段都包含正确的信息。您可以在 Amazon Pinpoint 控制台中更改您的公司注册的某些部分。有关修改公司注册详细信息的更多信息，请参阅编辑 10 家DLC公司的注册。

正在审核您的 Amazon SNS 10 注册信息 DLC

如果您的公司注册成功，并且您想注册具有更高吞吐量的活动，必须审查公司注册。

审查注册时，第三方组织会分析您提供的公司详细信息并返回审查分数。较高的审核分数可以提高您的 10 DLC 家公司及其相关活动的吞吐率。但是，审查不能保证提高吞吐量。

审查分数不具有追溯性。换句话说，如果你已经创建了 10 个DLC广告系列，之后又审核了你的公司注册，那么你的审核分数不会自动应用于你现有的广告系列。因此，在创建 10 个DLC广告系列中的任何一个之前，您都应该对自己的公司或品牌进行审查。

> **Note**
>
> 审查您的公司或品牌需支付 40 美元，且费用不可退还。

审查您的公司注册

1. 登录 AWS Management Console 并打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的 “SMS和语音” 下，选择 “电话号码”。
3. 在 10 个DLC广告系列选项卡上，选择要审查的 10 DLC 家公司。
4. 在公司详细信息页面上，选择页面底部的申请审查。
5. 在申请额外审查窗口中，选择提交。

   对于美国公司，审查过程通常需要大约一分钟即可完成。对于非美国公司，审查过程可能需要较长时间，具体取决于相关国家/地区的数据准备情况。

提交审查申请后，返回公司详细信息页面。公司审查结果部分会显示审查请求的状态和结果。审查过程完成后，此表中的分数列会显示得分。您的审核分数决定了您的 10 个DLC吞吐能力。您的吞吐量因您创建的活动类型而异。如果您创建了混合用途或与营销相关的 10 个DLC广告系列，则需要获得比其他广告系列类型更高的审核分数，才能实现高吞吐率。有关 10 个DLC电话号码功能的更多信息，请参阅10 种DLC能力。

如果您在完成审核过程后更改了公司注册的详细信息，可以请求再次审核注册。如果您只是更改了公司注册的“垂直行业”，那么您的审查分数不会变化。如果您更改了“垂直行业”以外的任何细节，则您的审查结果可能会发生变化。无论哪种情况，都会再次向您收取一次性审查费用。

编辑或删除 Amazon SNS 10 DLC 注册公司

您可以直接在亚马逊 Pinpoint 控制台中编辑贵公司的 10 个DLC注册信息中的一部分。您也可以通过在 Su AWS pport Center 中创建案例来删除 10 DLC 家公司的注册。

编辑 10 家DLC公司的注册

完成公司的 10 个DLC注册流程后，您可以编辑注册的详细信息。

如果在编辑公司注册详细信息后看到错误消息，说明注册可能还存在其他问题。您可以向 Support AWS 提交工单以获取更多信息。

编辑公司注册

1. 打开 AWS SMS 控制台，网址为https://console.aws.amazon.com/sms-voice/。
2. 按照《AWS End User Messaging SMS 用户指南》中编辑注册的说明进行操作。

删除 10 DLC 公司注册

删除公司注册

1. 打开 AWS SMS 控制台，网址为https://console.aws.amazon.com/sms-voice/。
2. 按照《AWS End User Messaging SMS 用户指南》中有关删除注册的说明进行操作。

亚马逊 SNS 10 DLC 活动注册

当您注册 10 个DLC广告系列时，您需要提供您的用例描述以及您计划使用的消息模板。在创建和注册 10 DLC 广告系列之前，必须先注册您的公司。有关注册公司的信息，请参阅亚马逊 SNS 10 DLC 公司注册流程。

> **Note**
>
> 当您注册公司后，Amazon Pinpoint 会显示以下两种注册状态之一：已验证或未验证。只有当您的公司注册状态为 “已验证” 时，您才能完成 10 个DLC活动注册流程。您将能够创建少量混合用途活动。
> 如果状态为 Unverified（未验证），这通常意味着您在注册公司时提供的某些数据不正确。当您的公司处于此状态时，您将无法创建任意 10 个DLC广告活动。您可以修改公司注册以尝试解决公司注册中存在的问题。有关修改 10 个DLC公司注册的更多信息，请参阅编辑或删除 Amazon SNS 10 DLC 注册公司。

在此页面上，您首先提供有关您正在为其创建 10 个DLC广告系列的公司的详细信息，然后提供该活动本身的用例详细信息。此页面上的信息随后提供给活动注册处以供批准。

在本节中，您将选择要为其创建 10 个DLC广告系列的公司并提供更多详细信息。

要注册 10 个DLC广告系列

1. 登录 AWS Management Console 并打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在 “SMS和语音” 下，选择 “电话号码”。
3. 在 10 个DLC广告系列选项卡上，选择创建 10 个DLC广告系列。
4. 在 “创建 10 个DLC广告系列” 页面的 “广告活动信息” 部分，执行以下操作：
   a. 对于 Company name（公司名称），选择您要为其创建此活动的公司。如果您尚未注册公司，必须先注册。有关注册公司的更多信息，请参阅亚马逊 SNS 10 DLC 公司注册流程。
   b. 在 10 个DLC广告活动名称中，输入该活动的名称。
   c. 对于垂直行业，选择最能代表您的公司的选项。
   d. 在 “帮助消息” 中，输入客户向您的 10 个DLC电话号码发送关键字 HELP “” 时收到的消息。
   e. 在 “停止消息” 中，输入客户向您的 10 个DLC电话号码发送关键字 STOP “” 时收到的消息。

> **Tip**
>
> 您的客户可以用 “HELP” 一词回复您的消息，以详细了解他们从您那里收到的消息。他们还可以回复 “STOP”，选择不接收您的消息。美国移动运营商要求您对这两个关键字提供回复。
> 以下是符合美国移动运营商要求的HELP回复示例：

ExampleCorp Account Alerts: For help call 1-888-555-0142 or go to example.com. Msg&data rates may apply. Text STOP to cancel.
以下是合规STOP响应的示例：
You are unsubscribed from ExampleCorp Account Alerts. No more messages will be sent. Reply HELP for help or call 1-888-555-0142.
您对这些关键字的响应不能超过 160 个字符。

5. 在 Campaign use case（活动使用案例）部分，执行以下操作：

   a. 对于使用案例类型，如果您的用例与慈善事业相关，请选择特殊。否则，请选择标准。

   b. 对于使用案例，从预设使用案例列表中选择与您的活动最相似的使用案例。每个使用案例的月度费用将显示在使用案例名称旁边。

   > **Note**
   > 每种用例类型旁边都显示了注册 10 个DLC活动的月度费用。大多数 10 种DLC广告系列的月度费用相同。注册少量混合用途使用案例的费用低于其他使用案例类型。但是，少量混合用途活动支持的吞吐率比其他活动类型低。

   c. 至少输入一条示例SMS消息。该消息是您计划向客户发送的示例消息。如果您计划在这个 10 个DLC活动中使用多个消息模板，请同时包括这些模板。

   > **Important**
   > 请勿对示例消息使用占位符文本。您提供的示例消息应尽可能准确地反映您计划发送的实际消息。

6. 活动和内容属性部分包含一系列与活动的特定功能相关的是或否问题。有些属性是必需的，因此您无法更改默认值。

   确保您选择的属性对于您的活动是准确的。

   指明以下各项是否适用于您正在注册的活动：

   - Subscriber opt-in（订阅者选择加入）– 订阅者可以选择接收有关此活动的消息。
   - Subscriber opt-out（订阅者选择退出）– 订阅者可以选择不接收有关此活动的消息。
   - 订阅者帮助-订阅者可以在发送HELP关键字后联系消息发件人。

- 号码共享 — 这 10 个DLC广告系列使用超过 50 个电话号码。
- 直接借贷或贷款安排 – 此活动包括直接借贷或其他贷款安排的信息。
- 嵌入式链接 — 10 DLC 广告系列包括嵌入式链接。不允许来自常用URL缩短器（例如 TinyUrl 或 Bit.ly）的链接。但是，您可以使用提供自定义URL域名的缩短器。
- Embedded phone number（嵌入式电话号码）– 活动包括一个嵌入式电话号码，该号码不是客户支持号码。
- 联盟营销 — 10 DLC 活动包括来自联盟营销的信息。
- 年龄分级内容 — 10 个DLC广告系列包括运营商和蜂窝电信与互联网协会 (CTIA) 指南中定义的按年龄划分的内容。

7. 选择创建。

   提交广告活动的注册详细信息后，将打开SMS和语音页面。此时将显示一条消息，指示您的活动已提交并且正在审查中。您可以在 10 个DLC广告系列选项卡上查看您的请求状态。您可以在 10 DLC 选项卡上查看注册状态，该选项卡将是以下选项之一：

   - 激活-您的 10 个DLC广告活动已获得批准。您可以申请一个 10 的DLC电话号码，并将该号码与您的广告活动相关联。有关更多信息，请参阅 请求 10 个DLC号码、免费电话号码和 P2P 长码，以便SMS通过 Amazon 发送消息 SNS。
   - 待处理-您的 10 个DLC广告活动尚未获得批准。在某些情况下，批准可能需要一周或更长时间。如果状态更改，Amazon Pinpoint 控制台会反映此更改。我们不会通知您状态更改。
   - 已拒绝-您的 10 个DLC广告活动已被拒绝。要获取更多信息，请提交支持请求，并在其中说明被拒绝活动的活动 ID。
   - 已暂停-一家或多家运营商暂停了您的 10 个DLC广告活动。要获取更多信息，请提交支持请求，并在请求中包括被暂停活动的 ID。Amazon Pinpoint 不会在控制台中包括暂停原因，如果您的活动被暂停，我们也不会通知您。

8. 如果您DLC的 10 获得批准，则可以申请一个 10 的DLC号码与该活动关联。有关请求 10 DLC 号码的信息，请参阅请求 10 个DLC号码、免费电话号码和 P2P 长码，以便SMS通过 Amazon 发送消息 SNS。

在多个 AWS 区域使用 10 个DLC广告活动

当您注册一家公司时，该公司将出现在您的所有 AWS 地区的 AWS 账户中。但是，对于 10 个DLC广告系列，情况并非如此。10 个DLC广告系列只能在其注册的 AWS 地区使用。

如果您计划DLC在多个 AWS 地区使用 10，则必须在每个区域分别注册 10 个DLC广告活动。为了遵守运营商要求，必须执行此步骤。即使使用案例完全相同，也需要为注册的每个活动付费。

注册多个活动还有一个额外的好处，那就是可以提高发送给使用AT＆T作为移动运营商的收件人的消息的吞吐率，因为AT＆T为每个活动提供10个DLC吞吐率。将其与T-Mobile处理10个DLC吞吐量的方式进行比较，后者基于每家公司的每日消息分配（无论活动数量多少）。

在 Amazon 上编辑或删除 10 个DLC广告活动 SNS

您可以使用 Amazon Pinpoint 控制台编辑 10 个DLC活动的STOP回复、回复和示例消息。HELP您也可以使用控制台删除 10 个DLC广告系列。

编辑 10 个DLC广告系列

活动获得批准后，您可以修改HELPSTOP、和示例消息。还可以添加其他示例消息。对这些字段的更改无需获得 Campaign Registry 或运营商的重新批准。10 DLC 广告系列获得批准后，您无法修改任何其他字段。

您最多可以拥有五条示例消息。您无法减少最初注册的示例消息数量。例如，如果您在活动中注册了三SMS条示例消息，则无法将示例SMS消息的数量减少到三条以下。

> **Note**
>
> 如果要修改除HELP、和示例消息之外的任何字段STOP，则必须先删除 10 个DLC广告系列，然后重新创建广告系列以包含更新的信息。

编辑 10 DLC 广告系列

1. 登录 AWS Management Console 并打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的 “SMS和语音” 下，选择 “电话号码”。
3. 在 10 个DLC广告系列选项卡上，选择要编辑的 10 个DLC广告活动。
4. 在活动详细信息页面的活动消息部分，选择编辑。
5. 更新以下任何字段：

   - Help message（HELP 消息）
   - 停止消息
   - 示例SMS消息

   您不能删除之前添加的示例消息，也不能删除示例消息的内容以使该字段为空。如果在不替换消息内容的情况下删除消息内容，将在更新时使用原始消息。

6. 选择更新。此时将显示一条确认横幅，告诉您活动消息已更新。

删除 10 DLC 广告系列

您可以使用亚马逊 Pinpoint 控制台删除 10 个DLC广告系列。在删除 10 DLC 广告系列之前，必须先删除与该活动关联的所有电话号码。

> **Important**
>
> 当你从活动中移除 10 DLC 号码时，你将无法再访问该号码。此外，已删除的 10 个DLC广告活动无法恢复。

删除 10 个DLC广告系列

1. 登录 AWS Management Console 并打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的 “SMS和语音” 下，选择 “电话号码”。
3. 在 10 个DLC广告系列选项卡上，选择要编辑的 10 个DLC广告活动。
4. 在电话号码部分，记下与活动关联的电话号码。
5. 在 “电话号码” 选项卡上，选择要删除的 10 个DLC号码，然后选择 “删除电话号码”。

   > **Note**
   >
   > 只有在您有多个 10 个DLC电话号码与活动关联时，才需要执行此步骤。如果您只有一个电话号码与 10 个DLC广告系列相关联，则该号码将显示在 10 个DLC广告系列选项卡上。记下选项卡上显示的号码。

6. 在确认框中输入 **`delete`**，然后选择 Confirm（确认）。成功消息显示在SMS和语音页面的顶部。
7. 对与活动关联的每个 10 个DLC号码重复前两个步骤。
8. 删除与 10 个DLC广告系列关联的所有数字后，选择 10 个DLC广告系列选项卡。
9. 选择您要删除的 10 个DLC广告系列。
10. 在 10 个DLC广告系列详情页面的右上角，选择删除。
11. 在确认框中输入 **`delete`**，然后选择 Confirm（确认）。成功消息显示在SMS和语音页面的顶部。

在 Amazon 中将长代码与 10 DLC 广告系列关联 SNS

如果您已有长代码，则可以通过提交支持请求将该长代码与当前 10 个DLC广告活动中的一个关联起来。您与 10 DLC 广告系列关联的长代码只能用于该广告系列，不能用于任何其他 10 个DLC广告系列。当你的长代码迁移到 10 DLC 时，你仍然可以使用它。但是，在获得批准之前，你将无法将其用于任何 10 个DLC广告系列。

当提交请求时，您需要：

- 与 10 DLC 广告系列关联的长代码
- 要与长代码关联的 10 个DLC广告系列 ID

> **Note**
>
> 在将任何长代码与活动关联之前，您需要先注册这 10 个DLC广告活动。如果您尚未创建和注册 10 个DLC广告系列，请参阅亚马逊 SNS 10 DLC 活动注册。

将长代码分配给 10 DLC

1. 登录 AWS Management Console 并打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在 “设置” 下，SMS然后在 “语音” 下，选择 “电话号码” 选项卡。
3. 选择要转换为 10 的长码DLC。
4. 要打开 Support Center，请选择 “分配给 10” DLC 活动。
5. 对于案例类型，请选择 Service limit increase（提高服务限制）。
6. 对于 Limit type（限制类型），选择 Pinpoint。
7. 在 “请求” 部分中，选择区域，然后在 “限制” 中选择 “10 DLC-将现有的美国长码关联到 10” DLC 活动。
8. 在 “案例描述” 下，在用例描述中，请务必包含 10 个DLC广告系列 ID 和要关联该广告系列的长代码。您可以在请求中包含多个长代码，但您应该只包含一个活动 ID。
9. 在 “联系人选项” 下，在 “首选联系语言” 中，选择您与 Su AWS pport 团队沟通时首选使用的语言。
10. 对于联系方式，请选择您首选的与 Su AWS pport 团队沟通的方法。
11. 选择提交。

亚马逊 SNS 10 DLC 跨账户访问权限

每 10 个DLC电话号码与一个 AWS 区域中的一个账户相关联。如果您想使用相同的 10 个DLC电话号码向多个账户或地区发送消息，则有两种选择：

1. 您可以在每个 AWS 账户中注册相同的公司和活动。这些注册单独管理和收费。如果您在多个 AWS 账户中注册同一家公司，则您每天可以发送给T-Mobile客户的消息数量将在每个账户中共享。
2. 你可以在一个 AWS 账户中完成 10 的DLC注册流程，然后使用 AWS Identity and Access Management (IAM) 授予其他账户通过你的 10 DLC 号码发送的权限。

> **Note**
>
> 此选项允许真正的跨账户访问您的 10 个DLC电话号码。但是，请注意，从您的辅助账户发送的消息被视为从您的主账户发送的消息。限额和账单计入主账户，而不是任何辅助账户。

使用IAM策略设置跨账户访问权限

您可以使用IAM角色将其他账户与您的主账户关联。然后，您可以通过授予次要账户访问主账户中的 10 个DLC号码的权限，将主账户的访问权限委托给辅助账户。

授予访问主账户中的 10 DLC 号码的权限

1. 如果您尚未完成此操作，请在主账户中完成 10 个DLC注册流程。此流程包括三个步骤：
   - 注册您的公司。有关更多信息，注册您的公司或品牌请参见与 10 配合使用DLC。
   - 注册你的 10 个DLC广告系列（用例）。有关更多信息，请参阅 亚马逊 SNS 10 DLC 活动注册。
   - 将电话号码与您的 10 DLC 广告系列关联。有关更多信息，请参阅 在 Amazon 中将长代码与 10 DLC 广告系列关联 SNS。
2. 在您的主账号中创建一个IAM角色，允许其他账号使用您的 10 DLC 电话号码调用该`Publish`API接口。有关创建角色的更多信息，请参阅《IAM用户指南》中的创建IAM角色。
3. 使用IAM角色向需要使用您的 10 个DLC号码的任何其他账户委派和测试您的主账户的访问权限。例如，您可以将访问权限从生产账户委派到开发账户。有关委派和测试权限的更多信息，请参阅《IAM用户指南》中的使用IAM角色委派跨 AWS 账户访问权限。

4. 使用新角色，使用主账DLC号中的 10 号码发送消息。有关使用角色的更多信息，请参阅IAM用户指南中的使用IAM角色。

亚马逊 SNS 10 DLC 注册问题解决方案

在某些情况下，当你尝试注册公司或 10 个DLC广告系列时，你可能会收到错误消息。

公司注册问题

注册公司时，您会看到以下两种注册状态之一：已验证或未验证。如果公司注册状态为已验证，则表示您的公司注册成功。您可以开始创建 10 个DLC广告系列。

如果公司注册状态为 Unverified（未验证），说明您提供的信息有问题。Amazon Pinpoint 控制台提供有关您的公司注册获得此状态的原因信息。

查看您的 10 DLC 家公司注册的注册问题

1. 登录 AWS Management Console 并打开亚马逊 Pinpoint 控制台，网址为。https://console.aws.amazon.com/pinpoint/
2. 在导航窗格的下方 SMS，选择电话号码。
3. 在 10 个DLC广告系列选项卡的广告系列列表中，选择要查找更多信息的公司名称。
4. 公司详细信息页面中包含有关注册中发现的问题的信息。如果 Company info（公司信息）部分中的字段包含警告符号，表示注册问题与该字段中的信息有关。

   请检查您提供的信息并确认所有字段都包含正确的信息。您可以在 Amazon Pinpoint 控制台中编辑公司注册。有关修改公司注册详细信息的更多信息，请参阅编辑或删除 Amazon SNS 10 DLC 注册公司。

活动注册问题

当你注册 10 个DLC广告系列时，在某些情况下你可能会看到一条错误消息。

如果您无法确定注册问题，可以使用 AWS Support Center 创建案例以请求更多信息。使用以下过程创建 Su AWS pport 案例。S AWS upport 团队将提供有关您的 10 个DLC活动注册被拒绝的原因的信息。

提交请求以获取有关已拒绝的 10 个DLC广告活动的信息

1. 登录 AWS Management Console 到 https://console.aws.amazon.com/。

2. 在 Support (支持) 菜单上，选择 Support Center (支持中心)。
3. 在您的支持案例窗格上，选择创建案例。
4. 选择想要提高服务限制？链接，然后完成以下操作：
   - 对于限制类型，选择PinpointSMS。
5. 在请求下，填写以下部分：
   - 对于区域，选择您尝试注册广告活动所在的地区。 AWS 区域

   > **Note**
   > 请求部分中必须填写“区域”。即使您在案例详情部分中提供了这些信息，也必须在此处包含这些信息。

   - 对于资源类型，选择 10 DLC 注册。
   - 在 “限制” 中，选择 “公司” 或 “10 次DLC活动注册被拒绝”。
6. 对于新限制值，选择限制类型的上限。通常，此值为 **1**。
7. 在案例描述下，输入被拒绝的 10 个DLC广告系列 ID。
8. （可选）如果您想提交其他任何请求，请选择添加其他请求。如果包含多个请求，请提供每个请求所需的信息。有关所需信息，请参阅为使用 Amazon SNS 进行 SMS 消息收发请求支持内的其他部分。
9. 在联系选项下，对于首选联系语言，请选择您希望接收有关此案例的通信时使用的语言。
10. 完成后，选择 Submit（提交）。

Amazon SNS 免费电话号码

免费电话号码 (TFN) 是一个 10 位数的号码，以以下区号之一开头：800、888、877、866、855、844 或 833。您只能TFNs使用发送交易消息。

> **Important**
> 美国移动运营商最近修改了法规，要求所有免费电话号码 (TFNs) 在2022年9月30日之前向监管机构完成注册程序。请前往，查看您的TFN状态the section called “免费电话号码注册状态”。有关注册您的公司的更多信息，请参阅the section called “注册免费电话号码”。
> 提交注册后，可能需要多达 15 个工作日才能得到处理。

2023 年 3 月 3 日更新：自 2023 年 4 月 1 日起，对于通过任何未注册免费电话号码发送的消息，移动运营商将采用以下全行业通用的阈值：

- 每日上限：500 条消息，在太平洋时间午夜 12:00 重置
- 每周限制：1000 条消息，在太平洋时间周日午夜 12:00 重置
- 每月上限：2000 条消息，在日历月底太平洋时间午夜 12:00 重置

2022 年 9 月 19 日更新：自 2022 年 10 月 1 日起，对于通过任何未注册免费电话号码发送的消息，移动运营商将采用以下全行业通用的阈值：

- 每日限额：2000 条消息
- 每周限额：12000 条消息
- 每月限额：25000 条消息

我们强烈建议您尽快完成注册。通过未注册发送的消息TFNs将尽力而为。随着运营商对未注册流量的限制日趋严格，这些消息逐渐会面临更多的筛选和屏蔽。

主题

免费电话号码使用指南

TFNs通常仅在美国境内用于交易消息，例如注册确认或发送一次性密码。它们既可用于语音留言，也可用于传送消息SMS。平均吞吐量为每秒三个消息部分 (MPS)。但是，此吞吐量受字符编码的影响。有关字符编码如何影响消息分段的更多信息，请参阅SMSAmazon 中的字符限制 SNS。有关注册的更多信息TFN，请参阅免费电话号码注册要求和流程。

每个客户账户最多可以有五个TFNs。但是，承运人可能需要正当的理由，允许同一用例最多五个TFNs，具体取决于其具体的批准流程和政策。如果您每秒发送的短信超过 15 条但少于 100 条，我们建议您注册一条或多条 10 条DLC发件IDs人。如果您的使用案例要求每秒发送 100 条以上的短信，我们建议您购买并注册一个或多个短代码。

使用TFN作为发件编号时，请遵循以下准则：

- 不要使用由第三方缩短URL器URLs创建的缩短邮件，因为这些邮件更有可能被过滤为垃圾邮件。

  如果您需要使用缩短的URL，请考虑使用 10 DLC 数字或短代码。使用短代码和 10 DLCs 需要注册消息模板，可以在其中指定缩短的消息模板URL。
- 请注意，关键词 opt-out (STOP) 和 opt-in (UNSTOP) 响应是在运营商层面设置的。您不能修改这些关键词或其他任何关键词。您也无法修改用户使用STOP和回复时发送的消息UNSTOP。
- 不要使用多条发送相同或相似的消息内容TFNs。运营商称这种做法为雪地行走或号码池并针对这些消息进行筛选。
- 与以下行业相关的任何消息都可能被视为受限消息，并受到严格过滤或被完全屏蔽。这可能包括与受限类别相关的服务的一次性密码 (OTPMFA) 和多因素身份验证 ()。

  如果您的注册因不合规用例而被拒绝，并且您认为此决定不正确，则可以通过支持部门提交申请。有关如何执行该操作的详细信息，请参阅 注册问题。

  下表描述受限内容的类型：

类别	示例
赌博	• 应用程序/网站 • 赌场 • 抽奖活动
高风险金融服务	• 汽车贷款 • Cryptocurrency • 收债 • 发薪日贷款 • 短期高息贷款 • 按揭贷款 • 学生贷款

类别	示例
	• 股市提醒
债务豁免	• 债务重整 • 债务减免 • 信用修复计划
G et-rich-quick 方案	• W ork-from-home 节目 • 风险投资机会 • 金字塔或多层次营销计划
禁用/受管制物质	• 大麻/ CBD
网络钓鱼	• 试图让用户透露个人信息或网站登录信息
S.H.A.F.T。	• 性 • 仇恨 • 酒精 • 枪支 • 烟草/电子烟

购买免费电话号码

要购买TFNs，请使用位于的亚马逊 Pinpoint 控制台。https://console.aws.amazon.com/sms-voice/有关更多信息，请参阅 免费电话号码注册要求和流程。

目前， AWS End User Messaging SMS 支持免费拨打语音和SMS留言号码。Amazon 仅SNS支持SMS消息传送。

免费电话号码注册要求和流程

> **Important**
>
> 如果TFN将 A 用于其指定用例以外的任何用途，则可以将其撤销。

免费电话号码禁止使用案例

在邮件被屏蔽（例如，与受控物质或网络钓鱼相关的用例）或预计需要进行高水平筛选（例如高风险金融消息）的情况下，Amazon SNS 发送消息的能力有限。您可能无法注册与中定义的受限内容用例TFNs相关联免费电话号码使用指南。

注册免费电话号码

购买后TFN，您必须注册该号码。有关如何执行此操作的说明，请参阅《AWS End User Messaging SMS 用户指南》中的免费电话号码注册流程。

自助注册各地区的免费电话号码 AWS End User Messaging SMS

如果您已TFN在各AWS End User Messaging SMS 地区申请，请使用《AWS End User Messaging SMS 用户指南》中美国免费号码注册表中的说明，直接在AWS End User Messaging SMS 控制台中完成公司注册流程。

注册时TFN，请确保信息完整准确，否则您的注册可能会被拒绝。您输入的信息应与您公司的总部完全匹配。

地区以外地区的免费电话号码的手动表单注册流程 AWS End User Messaging SMS

1. 下载此 US_ TFN _Registration.zip 并使用示例注册表（AWS 美国免费电话注册表-Business-Final.docx）填写TFN注册CSV文件中的所需信息 (-Final.csv)。bulkUStfn

   每个注册请求或用例最多只能有五个TFNs。如果您认为自己有资格获得此规则的豁免，请提供详细解释以供考虑。列出与注册或使用案例相关的所有电话号码。
2. 向 AWS Support 创建案例。将您填好的CSV文件附在案例中，然后提交TFN注册申请。
3. 选择创建案例，然后选择想要提高服务限额？
4. 对于限制类型，请选择SNS短信。
5. 对于资源类型，选择 10 DLC 或免费电话号码注册。
6. 附上 US_ TFN _注册文件并提交申请。

需要注意的关键点

1. 提交所有必填信息后，可能需要长达两周才能处理注册。如果信息缺失或不完整，注册过程将被延迟。如果您的注册被拒绝，我们将帮助您找出被拒绝的原因，并建议改进您的市场活动的方法，使其可以注册。

2. TFNs非常适合需要有限吞吐量的交易用例，例如多因素身份验证 (MFA)。每个账户每秒最多TFN可以发送三个短信，每个客户账户最多可以发送五个短信TFNs。如果您每秒发送的短信超过 15 部分但少于 100 条，我们建议您注册一个或多个亚马逊 SNS 10 DLC发件IDs人。如果您的使用案例要求每秒发送 100 条以上的短信，我们建议您购买并注册一个或多个短代码。有关更多详细信息，请参阅免费电话号码使用指南。

免费电话号码注册状态

要查看您的注册状态，请参阅《AWS End User Messaging SMS 用户指南》中的 “检查您的注册状态”。

编辑、放弃和删除您的注册

使用AWS End User Messaging SMS 用户指南执行以下任务：

- 编辑您的注册
- 放弃您的注册
- 删除您的注册
- 查看您的注册资源

注册问题

如果您的免费号码注册未被接受，您将看到一条消息，说明该号码被拒绝的原因。

要提交有关被拒绝的免费电话号码的信息请求，请执行以下操作：

1. 登录 AWS Management Console 到 https://console.aws.amazon.com/。
2. 在 Support (支持) 菜单上，选择 Support Center (支持中心)。
3. 在您的支持案例窗格上，选择创建案例。
4. 选择想要提高服务限制？链接，然后完成以下操作：
   - 对于 Limit type（限制类型），选择 PinpointSMS。
5. 在请求下，填写以下部分：
   - 区域表示您尝试注册活动的地方。

> **Note**
>
> 请求部分中必须填写“区域”。即使您在案例详情部分中提供了这些信息，也必须在此处包含这些信息。

- 对于资源类型，选择 10 DLC 或TFN注册。
- 对于限制，请选择公司或活动注册拒绝。

6. 对于新限制值，选择限制类型的上限。通常，此值为 **1**。
7. （可选）如果您想提交其他任何请求，请选择添加其他请求。有关所需信息，请参阅为使用 Amazon SNS 进行 SMS 消息收发请求支持内的其他部分。
8. 在案例描述下，输入被拒绝的免费电话号码。
9. 在联系选项下，对于首选联系语言，请选择您希望接收有关此案例的通信时使用的语言。
10. 完成后，选择 Submit（提交）。

免费电话号码常见问题解答

有关TFN注册过程的常见问题。

我目前是否拥有免费电话号码？

查看您是否拥有免费电话号码

- 在 https://console.aws.amazon.com/sms-voice/ 处打开 AWS End User Messaging SMS 控制台。
- 在导航窗格的下方 SMS，选择电话号码。
- 该TFN类型被列为免费电话。

我必须注册我的免费电话号码吗？

是。要继续使用TFN您目前拥有的，您必须在 2022 年 9 月 30 日之前对其进行注册。如果您要在 2022 年 9 月 30 日TFN之后购买新商品，则必须先注册后才能发送消息。

如何购买免费电话号码？

按照使用 AWS End User Messaging SMS 主机申请电话号码中的说明进行购买TFN.

如何注册我的免费电话号码？

按照上的说明the section called “注册免费电话号码”进行注册TFN.

我的免费电话号码的注册状态是什么？这是什么意思？

按照the section called “免费电话号码注册状态”中的说明检查您的注册和状态。

我需要提供哪些信息？

您需要提供公司的地址、业务联系人和用例TFN。您可以在the section called “注册免费电话号码”中找到所需信息。

如果我的注册遭到拒绝，该怎么办？

如果您的注册被拒绝，则状态将更改为 Requires Updates（需要更新）。要进行更新，请参阅the section called “编辑、放弃和删除您的注册”。

我需要哪些权限？

您用于访问 AWS End User Messaging SMS 控制台的IAM用户/角色必须具有 *“sms-voice:*”* 权限，否则你会收到拒绝访问的错误。

免费电话号码的优点和缺点

优点

免费电话的发起人MPS拥有更高的长代码和良好的送达率。

劣势

无法控制选择退出和选择加入，因为它们是在运营商层面进行管理的。

请勿在消息URLs中包含缩短的内容，也不要使用该号码发送促销信息。请改用 10 DLC 数字或短码。当您使用短代码或 10 DLC 号码时，您需要注册消息模板，该模板可以包含缩短的消息URLs，也可以是促销消息。有关短代码的更多信息，请参阅Amazon SNS 专用短码。有关 10 的更多信息DLC，请参阅亚马逊 SNS 10 DLC。

Amazon SNS 专用短码

短代码是比常规电话号码短的数字序列。例如，在美国和加拿大，标准电话号码（长代码）包含 11 位数，而短代码包含 5 位或 6 位数字。Amazon SNS 支持专用的短代码。

专用短代码

如果您向美国或加拿大的收件人发送大量SMS消息，则可以购买专用的短代码。与共享池中的短代码不同，专用短代码留给您独家使用。

优点

使用好记的短代码有助于建立信任。如果您需要发送敏感信息，例如一次性密码等，使用短代码来发送不失为一个好办法，因为您的客户可以快速确定消息是不是真的由您发出。

如果您正在开展新的客户获取活动，则可以邀请潜在客户向您的短代码发送关键字（例如，“发送短信FOOTBALL至10987获取足球新闻和信息”）。短代码比长代码更容易记住，也更容易让客户输入设备。通过减少客户在注册您的营销程序时遇到的麻烦，您可以提高您的营销活动的有效性。

因为新的短代码只有得到移动运营商批准后才能使用，所以移动运营商不大可能将发自批准的短代码的消息标记为非请求消息。

使用短代码发送SMS消息时，与使用其他类型的原始身份相比，每隔 24 小时可以发送更多的消息。换言之，发送限额更高。每秒钟也能发送更多消息。即，更高的发送率。

劣势

获取短代码需要付出额外成本，并且实现时间长。例如，在美国，每个短代码需支付650.00美元的一次性设置费 (USD)，外加每个短代码每月额外收取995.00美元的经常性费用。短代码在全部运营商网络上生效需要 8-12 周时间。要查找不同国家/地区或区域的价格和预置时间，请完成向 Amazon 申请SMS发送消息的专用短代码 SNS中描述的过程。

亚马逊SNS person-to-person 长码

Important

自 2023 年 8 月 31 日起，向美国及其领土（波多黎各、关岛、美属萨摩亚群岛和US维尔京群岛）发送SMS短信时需要使用诸如亚马逊 SNS 10 DLC号码或免费电话号码之类的专用号码。如果您使用美国作为这些区域的位置，您的长代码请求将被拒绝。

Important

自 2021 年 6 月 1 日起，美国电信提供商不再支持使用 person-to-person (P2P) 长码与美国目的地通信 application-to-person (A2P)。相反，您需要为这些消息使用其他类型的源 ID。有关更多信息，请参阅 亚马逊 SNS 10 DLC。

P2P 长代码是使用收件人所在国家/地区的号码格式的电话号码。P2P 长代码也称长号码或虚拟移动号码。例如，在美国和加拿大，P2P 长代码包含 11 位数：1 位国家代码，3 位地区代码，7 位电话号码。

有关请求 P2P 长代码的更多信息，请参阅 请求 10 个DLC号码、免费电话号码和 P2P 长码，以便SMS通过 Amazon 发送消息 SNS。

优点

专用 P2P 长代码仅供您的 Amazon SNS 账户使用，不会与其他用户共享。当您使用专用 P2P 长代码时，可以指定在发送每条消息时要使用哪个 P2P 长代码。如果向同一个客户发送多条消息，则可以确保每条消息像是发自同一个电话号码。因此，专用 P2P 长代码对于建立您的品牌或标识很有帮助。

劣势

到 US 目的地的 A2P 通信不支持 P2P 长代码。

如果您每天从一个专用 P2P 长代码发送数百条消息，则移动运营商可能会将您的号码认定为一个发送非请求消息的号码。一旦您的 P2P 长代码被标记，则您的消息可能无法送达收件人。

P2P 长代码的吞吐量也有限。最高发送率因国家/地区而异。如需更多信息，请联系 AWS Support。如果您计划发送大量SMS消息，或者计划以每秒大于一条消息的速率发送，则应购买专用的短代码。

有些运营商不允许您使用P2P长码发送A2P SMS 消息，包括在美国。A2P SMS 是在客户向应用程序提交手机号码时发送到该客户的移动设备的消息。A2P 消息为单向会话，例如营销消息、一次性密码和预约提醒等。如果您计划发送 A2P 消息，则应购买专用的短码（如果您的客户在美国或加拿大），或者使用发件人 ID（如果您的收件人所在的国家或地区支持发件人IDs）。

10 DLC 号码仅用于在美国境内发送消息。使用 10 DLC 数字要求您注册公司品牌以及要与该号码关联的广告系列。获得批准后，您可以在 Amazon Pinpoint 控制台的SMS语音页面上申请一个 10 的DLC电话号码，网址为。https://console.aws.amazon.com/pinpoint/申请后，获得批准的时间为 7-10 天。该号码不能与任何其他活动一起使用。

Amazon SNS 美国商品编号对比

此表显示了美国电话号码类型的支持比较。

产品功能	短代码	免费电话号码	10 DLC
号码格式	5-6 位数	10 位数字	10 位数字
渠道支持	SMS	SMS	SMS

产品功能	短代码	免费电话号码	10 DLC
SMS流量类型	促销和交易	交易	促销和交易
需要进行审查	是	否	是
估计预置时间	12 周[1]	15 个工作日	1 周
SMS吞吐量（每秒SMS消息数）[2]	每秒 100 个消息部分；更高的吞吐量需要支付额外费用。	每秒 3 个消息部分	根据您的 10 次DLC注册而有所不同。最多支持每秒 100 个消息部分。
所需关键字	选择加入、选择退出和 HELP	STOP，UNSTOP。这些均由网络托管。您无法更改选择退出并选择重新加入消息。	选择加入、选择退出和 HELP

[1] 预置估计不包括批准时间。

[2] 有关SMS邮件最大大小的更多信息，请参阅使用 Amazon 向手机发布SMS消息 SNS。

为使用 Amazon SNS 进行 SMS 消息收发请求支持

使用 Amazon SNS 的某些 SMS 选项在您联系 AWS Support 之前不适用于您的 AWS 账户。在 AWS Support 中心中创建一个案例来请求以下任意项目：

- 提高您的每月 SMS 支持阈值

  默认情况下，每月支出阈值设为 1.00 美元 (USD)。您的支出阈值决定您可以使用 Amazon SNS 发送的消息量。您可以为您的 SMS 使用案例请求符合预计的每月消息量的支出阈值。
- 从 SMS 沙盒迁移，以便您可以不受限制地发送 SMS 消息。有关更多信息，请参阅退出 Amazon SNS SMS 沙箱。
- 专用源号码
- 专用发送人 ID

发送人 ID 是一个自定义 ID，它在接收人的设备上显示为发送人。例如，您可以使用自己的企业品牌让消息来源更易于识别。不同国家或地区对发件人 ID 的支持有所不同。有关更多信息，请参阅Amazon SNS 支持的国家和地区。

您在 AWS Support 中心中创建案例时，请务必包括您提交的请求类型所需的所有信息。否则，AWS Support 必须联系您以获取此信息，然后再继续。通过提交详细案例，您可帮助确保案例即时完成。有关特定类型的 SMS 请求所需的详细信息，请参阅以下主题。

主题

向 Amazon 申请SMS发送消息的专用短代码 SNS

短代码是可用于发送大量SMS消息的数字。短代码通常用于 application-to-person (A2P) 消息、双因素身份验证 (2FA) 和营销。短代码通常包含 3 到 7 位数，具体取决于其所在的国家或区域。

您只能使用短代码将消息发送给位于短代码所在的同一国家/地区的接收人。如果您的使用情形要求您在多个国家/地区使用短代码，则您必须为您的接收人所在的每个国家/地区单独请求一个短代码。

有关短代码定价的信息，请参阅 Amazon SNS 定价。

> **Important**
>
> 如果您不熟悉通过 Amazon SMS 发送消息SNS，请申请一个符合您SMS用例预期需求的每月SMS支出门槛。默认情况下，您的每月支出限额为 1.00 美元 (USD)。您可以在包括您的短代码请求的相同支持案例中请求提高支出阈值。或者，您可以使用单独的案例。有关更多信息，请参阅 申请增加您在 Amazon 的月SMS度支出配额 SNS。
> 此外，如果您请求专用的短代码来发送将包含或可能包含 Protected Health Information（PHI）的消息，则应在提交支持案例时在案例描述中注明此目的，详情如下。

打开 Amazon SNS 短代码支持案例

AWS Support 通过完成以下步骤打开案例。

> **Note**
>
> 请求表中的某些字段将标记“可选”。但是，AWS Support 需要以下步骤中提到的所有信息，才能处理您的请求。如果您没有提供所有必需的信息，可能会在处理请求期间遇到延迟。

请求专用短代码

1. 转到 AWS 支持中心。
2. 登录 AWS Management Console 到 https://console.aws.amazon.com/。
3. 在 Support (支持) 菜单上，选择 Support Center (支持中心)。
4. 在您的支持案例窗格上，选择创建案例。
5. 选择想要提高服务限制？链接，然后完成以下操作：
   - 对于限制类型，选择 SNS 文本消息，然后完成以下操作：
   - （可选）要提供指向将要发送SMS消息的网站或应用程序的链接，请提供指向受众选择接收SMS消息的网站或应用程序名称的链接。
   - （可选）对于您计划发送什么类型的消息，选择您计划使用长代码发送的消息类型：
     - 一次性密码 – 提供您的客户用于向您的网站或应用程序进行身份验证的密码的消息。
     - 促销 – 宣传您的业务或服务的非关键性消息，如特别优惠或公告。
     - 事务性 – 为客户事务提供支持的重要信息性消息，如订单确认或账户提醒。事务性消息不得包含促销或营销内容。
   - （可选）对于您要从哪个 AWS 区域发送消息，请选择您要从哪个区域发送消息。
   - （可选）对于您计划向哪些国家/地区发送消息，请输入您计划向哪些国家或地区发送SMS消息。
   - （可选）在客户如何选择从您这里接收消息中，提供关于客户如何选择从您这里接收消息的描述。
   - （可选）在请提供您计划用于向客户发送消息的消息模板字段中，包括您将要使用的模板。
6. 在请求下，填写以下部分：
   - 对于该地区，请 AWS 区域 为您的短代码请求选择

> **Note**
>
> 请求部分中必须填写“区域”。即使您在案例详情部分中提供了这些信息，也必须在此处包含这些信息。

- 对于资源类型，选择专用SMS短代码。
- 对于限制，选择与您的使用案例最相似的选项。

7. 在 “新限制值” 中，选择您要请求IDs的发件人数量。通常，此值为 **1**。
8. （可选）如果您想提交其他任何请求，请选择添加其他请求。有关所需信息，请参阅为使用 Amazon SNS 进行 SMS 消息收发请求支持内的其他部分。
9. 在案例描述下，汇总您的使用案例，包括您的收件人注册使用短代码发送的消息的方式，然后提供以下信息：

   - 公司信息：
     - 公司名称
     - 公司邮寄地址
     - 您的请求的主要联系人的姓名和电话号码
     - 您公司的支持人员的电子邮件地址和免费电话号码
     - 公司税务 ID
     - 您的产品或服务的名称

   - 用户注册流程：
     - 公司网站，或您的客户将登录以接收来自您的短代码的消息的网站。
     - 用户将如何注册以接收来自您的短代码的消息。指定以下一个或多个选项：
       - **`Text messages`**
       - **`Website`**
       - **`Mobile app`**
       - **`Other`** 如需选择其他选项，请加以说明。
     - 在您的网站、应用程序或其他位置注册消息的选项的文本。
     - 您计划用于双向确认的消息的序列。执行以下所有操作：

1. 您计划在用户注册时发送的SMS消息。此消息要求用户同意周期性消息。例如ExampleCorp：回复YES以接收账户交易提醒。可能收取短信和数据费。
2. 您预计来自用户的选择加入响应。这通常是一个关键字，例如YES。
3. 当客户将此关键字发送到您的短代码时，您要发送的确认消息。例如：您现在已注册接收来自的账户提醒 ExampleCorp。可能收取短信和数据费。STOP要取消的短信或HELP获取信息。

- 您的消息的目的：
  - 您计划使用短代码发送的消息的目的。指定下列选项之一：
    - **Promotions and marketing**
    - **Location-based services**
    - **Notifications**
    - **Information on demand**
    - **Group chat**
    - **Two-factor authentication (2FA)**
    - **Polling and surveys**
    - **Sweepstakes or contests**
    - **Other** 如需选择其他选项，请加以说明。
  - 无论您计划将短代码用于发送您自己的业务以外的业务的促销消息还是市场营销消息。
  - 您是否计划使用短代码发送将包含或可能包含受保护的健康信息 (PHI) 的消息，该信息由《健康保险流通与问责法》(HIPAA) 及相关立法和法规所定义。

- 消息内容：
  - 当客户通过向您发送特定关键字选择加入您的消息时您计划发送的消息。指定该关键字和消息时要小心，因为变更此消息可能需要几周的时间。当我们创建您的短代码时，我们会向您使用短代码所在国家/地区的移动电话运营商注册此关键字和消息。您的消息可能类似于以下示例：欢迎来到 *ProductName* 警报！收取短信和数据费。*2* 每月留言。回复HELP寻求帮助，取消STOP订阅。
  - 当客户使用关键字回复您的消息时，您想要发送的回复HELP。此消息必须包含客户支持联系人信息。例如：*ProductName* 提醒：帮助，网址为 *example.com/help* 或者 *(800) 555-0199*。 消息和数据费率适用。*2* 每月留言。回复STOP取消。

- 当客户使用关键字回复您的消息时，您想要发送的回复STOP。此消息必须确认用户将不再接收来自您的消息。例如：您已取消订阅 *ProductName* 警报。不再发送消息。回复HELP寻求帮助或 *(800) 555-0199*.
- 您计划作为定期提醒发送、提醒用户已订阅您的消息的文本。例如：提醒：您已订阅来自ExampleCorp的账户提醒。可能收取短信和数据费。STOP要取消的短信或HELP获取信息。
- 您计划使用短代码发送的每个消息类型的示例。至少提供三个示例。如果您计划发送三种以上的消息，请提供所有消息类型的示例。

> **Important**
>
> 移动运营商需要我们提供上面列出的所有信息，以便配置短代码。在您提供所有此类信息之前，我们无法处理您的请求。

10. （可选）如果您想提交其他任何请求，请选择添加其他请求。如果包含多个请求，请提供每个请求所需的信息。有关所需信息，请参阅为使用 Amazon SNS 进行 SMS 消息收发请求支持内的其他部分。
11. 在联系选项下，对于首选联系语言，请选择您希望接收有关此案例的通信时使用的语言。
12. 完成后，选择 Submit（提交）。

在收到您的请求后，我们将在 24 小时内提供初始响应。我们可能会与您联系，要求您提供更多信息。如果能够为您提供短代码，我们将为您发送有关您在请求中指定的国家或区域中获取短代码相关费用的信息。此外，我们还会估计在您所在的国家或区域预置短代码所需的时间量。预置短代码一般需要数周的时间，但此延迟可能长得多或短得多，具体取决于短代码所在的国家或区域。

> **Note**
>
> 在我们向运营商发起您的短代码请求之后，将会立即产生与使用短代码相关的费用。即使短代码尚未完全预置好，您也需要负责支付这些费用。

为了防止我们的系统被用于发送未经请求或恶意的内容，我们必须仔细审查每个请求。如果您的使用案例与我们的政策不符，我们可能无法准予您的请求。

后续步骤

您已向无线运营商注册了短代码，并在亚马逊SNS控制台中查看了您的设置。现在，您可以使用 Amazon SMS 发送SNS以短代码作为发件号的消息。

请求 10 个DLC号码、免费电话号码和 P2P 长码，以便SMS通过 Amazon 发送消息 SNS

> **Important**
>
> 自 2021 年 6 月 1 日起，美国电信提供商不再支持使用 person-to-person (P2P) 长码与美国目的地通信 application-to-person (A2P)。相反，您需要为这些消息使用其他类型的源 ID。有关更多信息，请参阅 亚马逊 SNS 10 DLC。

要请求 10 个DLC号码、免费电话号码和 P2P 长码，请使用控制台。 AWS End User Messaging SMS 有关详细说明，请参阅《AWS End User Messaging SMS 用户指南》中的申请号码。

> **Important**
>
> 美国移动运营商最近修改了法规，并将要求所有免费电话号码 (TFNs) 在2022年9月30日之前在监管机构完成注册程序。有关注册免费电话号码的更多信息，请参阅注册免费电话号码。如果您在 2022 年 9 月 30 日或之前购买了免费电话号码，其状态将为活动状态，直至 2022 年 10 月 1 日，除非您已完成注册并且该号码已返回（状态设置为已完成）。否则，它将处于挂起状态，在您注册号码、返回注册或注册设置为活动状态之前，您将无法使用它发送消息。注册可能最多需要 15 个工作日。

> **Note**
>
> 如果您不熟悉使用 Amazon SMS 发送消息SNS，则还应申请一个符合您SMS用例预期需求的每月SMS支出门槛。默认情况下，您的每月支出限额为 1.00 美元 (USD)。有关更多信息，请参阅 申请增加您在 Amazon 的月SMS度支出配额 SNS。

请求发件人IDs向 Amazon 发送SMS消息 SNS

> **Important**
>
> 如果您不熟悉通过 Amazon SMS 发送消息SNS，请申请一个符合您SMS用例预期需求的每月SMS支出门槛。默认情况下，您的每月支出限额为 1.00 美元 (USD)。您可以在包括您的发件人 ID 请求的相同支持案例中请求提高支出阈值。或者，如果您愿意，您可以开立一个单独的案例。有关更多信息，请参阅 申请增加您在 Amazon 的月SMS度支出配额 SNS。

在SMS消息传送中，发件人 ID 是在收件人设备上显示为消息发件人的姓名。发件人IDs是向邮件收件人表明自己身份的有用方法。

对发件人的支持因国家而异。例如，美国的运营商根本不支持发件人IDs，但印度的运营商要求发件人使用发件人IDs。有关支持发件人的完整列表IDs，请参阅Amazon SNS 支持的国家和地区。

> **Important**
>
> 某些要求您在使用发件人发送消息IDs之前先注册发件人。根据所在的国家，此注册过程可能需要几周。需要预先注册发件人的显示IDs在 “支持的国家 /地区” 页面的表格中。
> 您可以在多个 AWS 账户中为SMS消息使用和注册相同的发件人 ID。如果您具有企业支持并且要跨多个账户注册多个模板，请按照以下步骤操作，并与您的技术客户经理合作，确保您的注册体验协调一致。

如果您向支持发件人的国家收件人IDs发送邮件，并且该国家不要求您注册发件人 ID，则无需执行任何其他步骤。您可以立即开始发送包含发件人 ID 值的消息。

如果您计划向需要注册发件人的国家发送邮件，IDs则只需完成此页面上的程序。

第 1 步：打开亚马逊SNSSMS案例

如果您计划向需要发件人的国家/地区的收件人IDs发送消息，则可以通过在 Su AWS pport Center 中创建新问题来申请发件人 ID。

> **Note**
>
> 如果您计划向允许但不要求发件人的国家/地区的收件人IDs发送消息，则无需在 Support Center 中提交案例。您可以IDs立即开始发送使用发件人的消息。

请求发件人 ID

1. 登录 AWS Management Console 到 https://console.aws.amazon.com/。
2. 在 Support (支持) 菜单上，选择 Support Center (支持中心)。
3. 在您的支持案例窗格上，选择创建案例。
4. 选择想要提高服务限制？链接，然后完成以下操作：

   - 对于 Limit type（限制类型），选择 PinpointSMS。
   - （可选）要提供指向将要发送SMS消息的网站或应用程序的链接，请标识受众选择接收您的SMS消息的网站或应用程序。
   - （可选）对于您计划发送什么类型的消息，选择您计划使用长代码发送的消息类型：
     - 一次性密码 – 提供您的客户用于向您的网站或应用程序进行身份验证的密码的消息。
     - 促销 – 宣传您的业务或服务的非关键性消息，如特别优惠或公告。
     - 事务性 – 为客户事务提供支持的重要信息性消息，如订单确认或账户提醒。事务性消息不得包含促销或营销内容。
   - （可选）对于您要从哪个 AWS 地区发送消息，请选择您要从哪个地区发送SMS消息。 AWS 区域
   - （可选）对于您计划将消息发送到的国家/地区，请输入您希望注册发送人 ID 的国家。对发件 IDs人和发件人 ID 注册要求的 Support 因国家而异。有关更多信息，请参阅 Amazon SNS 支持的国家和地区。

     如果国家列表超出此文本框允许的字符数，您可以改为改为在案例描述部分中列出国家。
   - （可选）在客户如何选择从您这里接收消息中，提供关于客户如何选择从您这里接收消息的描述。
   - （可选）在请提供您计划用于向客户发送消息的消息模板字段中，包括您将要使用的任何消息模板。
5. 在请求下，填写以下部分：

   - 对于区域，为您的发件人 ID 选择 AWS 区域。

   > **Note**
   >
   > 请求部分中必须填写“区域”。即使您在案例详情部分中提供了这些信息，也必须在此处包含这些信息。

- 对于资源类型，选择一般限制。
- 对于 “限制”，选择 “SMS生产访问权限”。

6. 在 “新限制值” 中，选择您要请求IDs的发件人数量。通常，此值为 **1**。
7. （可选）如果您想提交其他任何请求，请选择添加其他请求。有关所需信息，请参阅为使用 Amazon SNS 进行 SMS 消息收发请求支持内的其他部分。
8. 在案例描述下，对于使用情形描述，提供以下详细信息：

- 要注册的发件人 ID。
- 您计划用于SMS消息的模板。
- 计划每个月发送给每个接收人的消息数。
- 有关客户如何选择从您这里接收消息的信息。
- 公司或组织的名称。
- 与公司或组织关联的地址。
- 公司或组织所在的国家/地区。
- 公司或组织的电话号码。
- 贵公司或组织的网站。URL

9. 在联系选项下，对于首选联系语言，请选择您希望接收有关此案例的通信时使用的语言。
10. 完成后，选择 Submit（提交）。

在收到您的请求后，我们将在 24 小时内提供初始响应。我们可能会与您联系，要求您提供更多信息。如果能够为您提供发件人 ID，我们将向您发送一份预置该 ID 所需时间量的估算。

为了防止我们的系统被用于发送未经请求或恶意的内容，我们必须仔细审查每个请求。如果您的使用案例与我们的政策不符，我们可能无法准予您的请求。

第 2 步：在 Amazon SNS 控制台中更新您的SMS设置

在完成获取您的发件人 ID 的过程时，我们会对您的案例作出响应。收到此通知后，请完成本节中的步骤，将 Amazon 配置SNS为使用您的账户发送的所有邮件使用您的发件人 ID 作为默认发件人 ID。或者，您可以选择指定发布消息时要使用的发件人 ID。

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 “手机”，然后选择 “短信” (SMS)。
3. 在文本消息首选项部分中，选择编辑。

4. 在详细信息部分的默认发件人 ID 字段中，输入提供的发件人 ID，以用作来自您账户的所有消息的默认 ID。
5. 完成后，选择 Save changes (保存更改)。

后续步骤

您已经在亚马逊SNS控制台中注册了发件人编号并更新了设置。现在，您可以使用 Amazon SNS 发送带有您的发件人 ID 的SMS消息。SMS受支持国家/地区的收件人将在其设备上将您的发件人 ID 视为消息发件人。如果在发布消息时使用不同的发件人 ID，它将覆盖此处配置的默认 ID。

申请增加您在 Amazon 的月SMS度支出配额 SNS

Amazon SNS 提供支出配额，以帮助您管理SMS使用您的账户进行汇款所产生的每月最高费用。支出配额限制了您在遭受恶意攻击时的风险，并防止上游应用程序发送超出预期的消息。您可以SNS将 Amazon 配置为在确定发送SMS消息所产生的费用超过您当月的支出配额时停止发布SMS消息。

为确保您的运营不受影响，我们建议申请足够高的支出配额，以支持您的生产工作负载。有关更多信息，请参阅步骤 1：提交 Amazon SNS SMS 案例。收到配额后，您可以通过应用全部配额或较小的值来管理风险，如步骤 2：更新SMS设置中所述。通过应用一个较低的值，您可以控制您的每月支出，并在必要时选择进行纵向扩展。

> **Important**
>
> 由于 Amazon SNS 是一个分布式系统，因此如果超过支出配额，它会在几分钟内停止发送SMS消息。在此期间，如果您继续发送SMS消息，则可能会产生超出配额的费用。

我们将所有新账户的支出配额设置为每月 1.00 美元 (USD)。此配额旨在让您测试 Amazon 的消息发送能力。SNS要申请增加账户的SMS支出配额，请在 Support Center 中提交增加配额的 AWS 案例。

第 1 步：打开亚马逊SNSSMS案例

您可以在 Support Center 中提交配额增加案例，申请增加每月支出 AWS 配额。

> **Note**
>
> 请求表中的某些字段将标记“可选”。但是， AWS Support 需要以下步骤中提到的所有信息，才能处理您的请求。如果您没有提供所有必需的信息，可能会在处理请求期间遇到延迟。

1. 登录 AWS Management Console 到 https://console.aws.amazon.com/。
2. 在 Support (支持) 菜单上，选择 Support Center (支持中心)。
3. 在您的支持案例窗格上，选择创建案例。
4. 选择想要提高服务限制？链接，然后完成以下操作：
   - 对于限制类型，选择 SNS 文本消息。
   - （可选）要提供指向将要发送SMS消息的网站或应用程序的链接，请提供有关将发送SMS消息的网站、应用程序或服务的信息。
   - （可选）对于您计划发送什么类型的消息，选择您计划使用长代码发送的消息类型：
     - 一次性密码 – 提供您的客户用于向您的网站或应用程序进行身份验证的密码的消息。
     - 促销 – 宣传您的业务或服务的非关键性消息，如特别优惠或公告。
     - 事务性 – 为客户事务提供支持的重要信息性消息，如订单确认或账户提醒。事务性消息不得包含促销或营销内容。
   - （可选）对于您要从哪个 AWS 区域发送消息，请选择您要从哪个区域发送消息。
   - （可选）对于您计划将消息发送到的国家/地区，输入您要在其中购买短代码的国家或地区。
   - （可选）在您的客户如何选择接收您的消息中，提供有关您的选择加入流程的详细信息。
   - （可选）在请提供您计划用于向客户发送消息的消息模板字段中，包括您将要使用的模板。
5. 在请求下，填写以下部分：
   - 对于区域，选择您要从中发送消息的区域。

     > **Note**
     >
     > 请求部分中必须填写“区域”。即使您在案例详情部分中提供了这些信息，也必须在此处包含这些信息。

   - 对于资源类型，选择一般限制。
   - 对于限制，选择提高账户支出阈值。
6. 在新的限额值中，输入SMS每个日历月可以花费的最大金额（英寸USD）。
7. 在案例描述下，对于使用情形描述，提供以下详细信息：
   - 发送SMS消息的公司或服务的网站或应用程序。
   - 您的网站或应用程序提供的服务，以及您的SMS消息如何为该服务做出贡献。

- 用户如何注册以自愿在您的网站、应用程序或其他位置接收您的SMS消息。

如果您申请的支出配额（您为新配额值指定的值）超过 10,000 美元 (USD)，请提供您要发送消息的每个国家/地区的以下其他详细信息：

- 您使用的是发件人 ID 还是短代码。如果使用的是发件人 ID，请提供：
  - 发件人 ID。
  - 此发件人 ID 是否已向该国家/地区的无线运营商注册。
- 您的消息传递的最大预期值 transactions-per-second (TPS)。
- 平均消息大小。
- 您将发送到该国家/地区的消息的模板。
- （可选）字符编码需求（如果有）。

8. （可选）如果您想提交其他任何请求，请选择添加其他请求。如果包含多个请求，请提供每个请求所需的信息。有关所需信息，请参阅为使用 Amazon SNS 进行 SMS 消息收发请求支持内的其他部分。
9. 在联系选项下，对于首选联系语言，请选择您希望接收有关此案例的通信时使用的语言。
10. 完成后，选择 Submit（提交）。

AWS Support 团队会在 24 小时内对您的请求做出初步回应。

为了防止我们的系统被用于发送未经请求或恶意的内容，我们要仔细考虑每个请求。如果我们可以，我们将在 24 小时内准予您的请求。但是，如果我们需要从您那里获得其他信息，则可能需要更长的时间来解决您的请求。

如果您的使用案例与我们的策略不符，我们可能无法准予您的请求。

第 2 步：在 Amazon SNS 主机上更新您的SMS设置

在我们通知您您的每月支出配额已增加后，您必须在 Amazon SNS 控制台上调整账户的支出配额。

> **Important**
>
> 您必须完成以下步骤，否则您的SMS支出限额将不会增加。

在控制台中调整您的支出限额

1. 登录 Amazon SNS 控制台。
2. 打开左侧导航菜单，展开 “手机”，然后选择 “短信” (SMS)。
3. 在移动短信 (SMS) 页面的短信偏好设置部分，选择编辑。
4. 在 “编辑短信偏好” 页面的 “详情” 部分，在 “账户SMS支出限制” 字段中输入新的支出限额。

> **Note**
>
> 您可能会看到一条警告，指出输入的值大于默认支出限额。您可以忽略此警告。

5. 选择 Save changes（保存更改）。

> **Note**
>
> 如果您收到 “参数无效” 错误，请查看 Su AWS pport 的联系并确认您输入了正确的新SMS支出限额。如果您仍然遇到问题，请在 Support Center 中 AWS 提交案例。

在 Amazon 中设置SMS消息偏好 SNS

使用 Amazon SNS 来指定SMS消息收发偏好。例如，您可以指定是否根据成本或可靠性来优化交付、您的每月支出限额、如何记录交付以及是否订阅每日SMS使用报告。

这些偏好设置对您从账户发送的每SMS条消息生效，但您可以在发送个人消息时覆盖其中的一些偏好。有关更多信息，请参阅 使用 Amazon 向手机发布SMS消息 SNS。

主题

- 使用设置SMS消息首选项 AWS Management Console
- 使用设置首选项 AWS SDKs
- 为特定国家/地区的配送设置SMS消息首选项

使用设置SMS消息首选项 AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 选择支持SMS消息传递的区域。
3. 在导航面板上，选择手机、短信 (SMS)。

4. 在移动短信 (SMS) 页面的短信偏好设置部分，选择编辑。
5. 在 Edit text messaging preferences (编辑文本消息发送首选项) 页上，在 Details (详细信息) 部分中，执行以下操作：

   a. 对于默认消息类型，选择下列选项之一：

      - 促销（默认）– 非重要消息（例如营销消息）。Amazon SNS 优化消息传送以实现最低成本。
      - 事务性（默认）– 为客户事务处理提供支持的重要消息，例如多重身份验证的一次性密码。Amazon SNS 优化消息传送以实现最高的可靠性。

      有关促销和交易消息的定价信息，请参阅全球SMS定价。

   b. （可选）在账户支出限额中，输入您想在每个日历月的SMS消息上花费的金额（单位 USD）。

      > **Important**
      >
      > - 默认情况下，支出配额设置为 1.00 USD。如果要提高服务配额，请提交请求。
      > - 如果控制台中设置的金额超过您的服务配额，Amazon 将SNS停止发布SMS消息。
      > - 由于 Amazon SNS 是一个分布式系统，因此它会在超过支出配额后的几分钟内停止发送SMS消息。在此间隔内，如果您继续发送SMS消息，则可能会产生超出配额的费用。

6. （可选）对于默认发件人 ID，请输入一个自定义 ID（如您的企业品牌），它显示为接收设备的发送者。

   > **Note**
   >
   > 对发件人的支持因国家/地区IDs而异。

7. （可选）输入 Amazon S3 bucket name for usage reports（使用情况报告的 Amazon S3 存储桶名称）的名称。

   > **Note**
   >
   > S3 存储桶策略必须授予对 Amazon 的写入权限SNS。

8. 选择 Save changes（保存更改）。

使用设置首选项 AWS SDKs

要使用其中一个设置您的SMS首选项 AWS SDKs，请使用与 SDK Amazon 中的`SetSMSAttributes`请求相对应的操作SNSAPI。通过此请求，您可以为不同的SMS属性分配值，例如您的每月支出配额和默认SMS类型（促销或交易）。有关所有SMS属性，请参阅《Amazon 简单通知服务API参考》etSMSAttributes中的 S。

以下代码示例演示如何使用 `SetSMSAttributes`。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

如何使用亚马逊SNS来设置 D efaultSMSType 属性。

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for C++ API参考资料》中的 S。

CLI

AWS CLI

设置SMS消息属性

以下set-sms-attributes示例将SMS邮件的默认发件人 ID 设置为MyName。

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

此命令不生成任何输出。

- 有关API详细信息，请参阅etSMSAttributes《AWS CLI 命令参考》中的 S。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for Java 2.x API参考资料》中的 S。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional – (Default) Noncritical messages, such as marketing
 messages.
```

```
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for JavaScript API参考资料》中的S。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for PHP API参考资料》中的 S。

为特定国家/地区的配送设置SMS消息首选项

您可以通过仅向特定目的地国家/地区发送消息来管理和控制SMS流量。这样可以确保您的邮件仅发送到经批准的国家/地区，从而避免不必要的SMS费用。以下说明使用 Amazon Pinpoint 的保护配置来指定您想要允许或阻止的国家/地区。

1. 打开 AWS SMS 控制台，网址为https://console.aws.amazon.com/sms-voice/。
2. 在导航窗格的 “概述” 下的 “快速入门” 部分，选择 “创建保护配置”。
3. 在 “保护配置详细信息” 下，为您的保护配置输入适合企业使用的名称（例如，仅允许-AU）。
4. 在SMS国家/地区规则下，选中 “地区/国家” 复选框以阻止向所有支持的国家/地区发送消息。
5. 取消选中您要向其发送消息的国家/地区的复选框。例如，要仅允许向澳大利亚发送消息，请取消选中澳大利亚的复选框。
6. 在 “保护配置关联” 部分的 “关联类型” 下，选择 “账户默认”。这将确保 Prot AWS End User Messaging SMS ect 配置影响通过亚马逊SNS、Amazon Cognito 和 Amazon Pinpoint 调用发送的所有消息。SendMessagesAPI
7. 选择 “创建保护配置” 以保存您的设置。

   将显示以下确认消息：

```
Success Protect configuration protect-abc0123456789 has been created.
```

8. 登录 Amazon SNS 控制台。
9. 向其中一个被封锁的国家@@ 发布消息，例如印度。

   该消息将不会被传送。您可以使用在传送失败日志中对此进行验证CloudWatch。搜索日志组 sns/region/accountID/ DirectPublishToPhoneNumber /Failure 以获得与以下示例类似的响应：

```
{
"notification": {
"messageId": "bd59a509-XXXX-XXXX-82f8-fbdb8cb68217",
"timestamp": "YYYY-MM-DD XX:XX:XX.XXXX“
},
"delivery": {
"destination": "+91XXXXXXXXXX",
"smsType": "Transactional",
"providerResponse": "Cannot deliver message to the specified destination country",
"dwellTimeMs": 85
},
"status": "FAILURE"
}
```

使用 Amazon 发送SMS消息 SNS

本节介绍如何使用 Amazon SNS 向主题和手机发送SMS消息。

主题

- 向 Amazon SNS 主题发布SMS消息
- 使用 Amazon 向手机发布SMS消息 SNS

向 Amazon SNS 主题发布SMS消息

您可以将一条SMS消息同时发布到多个电话号码，方法是将这些电话号码订阅到 Amazon SNS 话题。SNS主题是一种沟通渠道，您可以向其中添加订阅者，然后向所有这些订阅者发布消息。在您取消订阅或订阅者选择不接收来自您的 AWS 账户的消息之前，订阅者会收到发布到该主题的所有SMS消息。

主题

- 使用控制台向主题发送消息
- 使用向主题发送消息 AWS SDKs

使用控制台向主题发送消息

要创建主题，请执行以下操作

如果您还没有要向其发送SMS消息的主题，请完成以下步骤。

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，选择支持消息SMS传递的AWS 区域。
3. 在导航窗格中，选择 Topics（主题）。
4. 在 Topics（主页）页面上，选择 Create topic（创建主题）。
5. 在 Create topic（创建主题）页面上的 Details（详细信息）下，执行以下操作：

   a. 对于 Type（类型），选择 Standard（标准）。

   b. 对于 Name（名称），输入一个主题名称。

   c. （可选）在“显示名称”中，输入SMS消息的自定义前缀。当您向该主题发送消息时，Amazon 会在显示名称SNS前面加上一个右尖括号 (>) 和一个空格。显示名称不区分大小写，Amazon SNS 会将显示名称转换为大写字符。例如，如果主题的显示名称是 `MyTopic`，而消息是 `Hello World!`，则该消息会显示为：

   ```
   MYTOPIC> Hello World!
   ```

6. 选择创建主题。主题的名称和 Amazon 资源名称 (ARN) 显示在主题页面上。

创建订SMS阅

您可以使用订阅将一条SMS消息发送给多个收件人，方法是只向您的主题发布一次消息。

> **Note**
>
> 当您开始使用 Amazon SNS 发送SMS消息时，您的 AWS 账户处于SMS沙箱中。SMS沙盒为您提供了一个安全的环境，让您可以试用 Amazon 的SNS功能，而不必冒发件人声誉的SMS风险。当您的账户处于SMS沙箱状态时，您可以使用 Amazon 的所有功能SNS，但只能向经过验证的目标电话号码发送SMS消息。有关更多信息，请参阅 Amazon SNS SMS 沙箱。

1. 登录 Amazon SNS 控制台。
2. 在导航窗格中，选择订阅。
3. 在 Subscriptions（订阅）页面上，选择 Create subscription（创建订阅）。

4. 在 Create subscription（创建订阅）页面上的 Details（详细信息）下，执行以下操作：

   a. 在 “主题” 中ARN，输入或选择您要向其发送SMS消息的主题的 Amazon 资源名称 (ARN)。

   b. 对于 Protocol（协议），选择 SMS。

   c. 对于 Endpoint（终端节点），输入要订阅主题的电话号码。

5. 选择创建订阅。订阅信息显示在 Subscriptions（订阅）页面。

   要添加更多电话号码，请重复以下步骤。您还可以添加其他类型的订阅，例如电子邮件。

发送邮件

当您向某个主题发布消息时，Amazon SNS 会尝试将该消息传送到订阅该主题的每个电话号码。

1. 在 Amazon SNS 控制台的 “主题” 页面上，选择您要向其发送SMS消息的主题的名称。
2. 在主题详细信息页面上，选择发布消息。
3. 在 Publish message to topic（将消息发布到主题）页面上的 Message details（消息详细信息）下，执行以下操作：

   a. 对于 “主题”，除非您的主题包含电子邮件订阅并且您想同时发布到电子邮件和订阅，否则请将该字段留空。SMSAmazon SNS 使用您输入的主题作为电子邮件主题行。

   b. （可选）在 Time to Live (TTL) 中，输入亚马逊SNS向任何移动应用程序终端节点订阅者发送SMS消息所需的秒数。

4. 在 Message body（消息正文）下，执行以下操作：

   a. 对于 Message structure（消息结构），选择 Identical payload for all delivery protocols（所有传输协议的负载相同）向订阅了您的主题的所有协议类型发送相同消息。或者，选择 Custom payload for each delivery protocol（针对每个传输协议的自定义负载）为不同协议类型的订阅者自定义消息。例如，您可以输入电话号码订阅者的默认消息，以及电子邮件订阅者的自定义消息。

   b. 对于 Message body to send to the endpoint（要发送到终端节点的消息正文），输入您的消息或每个传输协议的自定义消息。

      如果您的主题有显示名称，Amazon SNS 会将其添加到消息中，这会增加消息长度。显示名称的长度是名称中的字符数加上右尖括号 (>) 的两个字符以及亚马逊SNS添加的空格。

      有关SMS消息大小配额的信息，请参阅使用 Amazon 向手机发布SMS消息 SNS。

5. （可选）在消息属性中，添加消息元数据，例如时间戳、签名和。IDs

6. 选择发布消息。Amazon SNS 发送SMS消息并显示成功消息。

使用向主题发送消息 AWS SDKs

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置AWS SDKs和凭据文件。

以下代码示例展示了如何：

- 创建 Amazon SNS 主题。
- 使用手机号码订阅主题。
- 向主题发布SMS消息，以便所有订阅的电话号码都能同时收到消息。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建一个主题并返回它ARN。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
}
```

为终端节点订阅主题。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives
 notifications (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

设置消息的属性，例如发件人的 ID、最高价格及其类型。消息属性是可选的。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

向主题发布消息。消息将会发送到每个订阅者。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();
```

```
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

使用 Amazon 向手机发布SMS消息 SNS

您可以使用 Amazon 将SMS消息直接发送SNS到手机，而无需将电话号码订阅到亚马逊SNS主题。

> **Note**
>
> 如果您想要将一条消息一次发布至多个电话号码，使用电话号码订阅主题会很有用。有关向主题发布SMS消息的说明，请参阅向 Amazon SNS 主题发布SMS消息。

在发送消息时，您可以控制是否对消息进行成本或可靠性传输优化。您还可以指定发件人 ID 或源号码。如果您使用 Amazon SNS API 或，以编程方式发送消息 AWS SDKs，则可以为消息传输指定最高价格。

每SMS条消息最多可包含 140 字节，字符配额取决于编码方案。例如，一条SMS消息可以包含：

- 160 GSM 个字符
- 140 ASCII 个字符
- 70 UCS -2 个字符

如果您发布的消息超过了大小配额，Amazon SNS 会将其作为多条消息发送，每条消息都符合大小配额。消息以整个词为边界，而不会在一个词的中间截断。单个SMS发布操作的总大小配额为 1,600 字节。

发送SMS消息时，您可以使用 E.164 格式指定电话号码，这是一种用于国际电信的标准电话号码结构。遵循此格式的电话号码最多可包含 15 位数字，并以加号 (+) 和国家/地区代码作为前缀。例如，E.164 格式的美国电话号码显示为 +1 XXX555 0100。

主题

- 使用控制台发送消息
- 使用发送消息 AWS SDKs

使用控制台发送消息

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，选择支持消息SMS传递的AWS 区域。
3. 在导航窗格中，选择短信 (SMS)。
4. 在移动短信 (SMS) 页面上，选择发布短信。
5. 在 “发布SMS消息” 页面上，对于 “消息类型”，选择以下选项之一：

   - Promotional（促销）– 不重要的消息，例如营销消息。
   - Transactional（事务性）– 为客户事务处理提供支持的重要消息，例如多重身份验证的一次性密码。

   > **Note**
   >
   > 此消息级别的设置会覆盖您的账户级别的默认消息类型。您可以从移动短信 (SMS) 页面的短信偏好设置部分设置账户级别的默认消息类型。

   有关促销和交易消息的定价信息，请参阅全球SMS定价。
6. 对于 Destination phone number（目标电话号码），请输入您想要向其发送消息的电话号码。
7. 对于 Message（消息），请输入要发送的消息。
8. （可选）在 Origination identities（源身份）下，指定如何向收件人识别自己：

   - 要指定 Sender ID（发件人 ID），请键入包含 3-11 个字母数字字符的自定义 ID，其中包括至少一个字母，并且不能包含空格。该发件人 ID 在接收设备上显示为消息发件人。例如，您可以使用自己的企业品牌让消息来源更易于识别。

对发件人的支持因国家和/或地区IDs而异。例如，发送至美国电话号码的消息不显示发件人 ID。有关支持发件人的国家和地区IDs，请参阅Amazon SNS 支持的国家和地区。

如果您未指定发件人 ID，以下标识之一将显示为源身份：

- 在支持长代码的国家/地区，显示长代码。
- 在仅支持发IDs件人的国家/地区 NOTICE，会显示。

此消息级别的发件人 ID 会覆盖您在 Text messaging preferences 页面中设置的默认发件人 ID。

- 要指定 Origination number（远啊号码），输入一个由 5-14 个数字组成的字符串，以显示为收件人设备上的发件人电话号码。此字符串必须与您在 AWS 账户 中为目的地国家/地区配置的起始号码相匹配。起始号码可以是 10 DLC 号码、免费电话号码、 person-to-person 长码或短码。有关更多信息，请参阅 Amazon SNS SMS 消息的来源身份。

  如果您未指定发件号，Amazon SNS 会根据您的 AWS 账户 配置选择用于SMS短信的发件号。

9. 如果您要向印度的收件人发送SMS邮件，请展开特定国家/地区的属性，并指定以下属性：

   - 实体 ID — 用于向印度收件人发送SMS消息的实体 ID 或委托人实体 (PE) ID。此 ID 是一个由 1-50 个字符组成的唯一字符串，由印度电信监管局 (TRAI) 提供，用于识别您在注册的实体。TRAI
   - 模板 ID — 用于向印度收件人发送SMS消息的模板 ID。此 ID 是一个由 1-50 个字符组成的唯一字符串，用于标识您在中注册的模板。TRAI TRAI模板 ID 必须与您为邮件指定的发件人 ID 相关联。

   有关向印度收件人发送SMS消息的更多信息，请参阅印度的 Amazon SNS 发件人 ID 注册要求。

10. 选择发布消息。

> **Tip**
>
> 要从发件号码发送SMS消息，您还可以在亚马逊SNS控制台导航面板中选择发件号码。选择“功能” 列SMS中包含的发件号码，然后选择 “发布短信”。

使用发送消息 AWS SDKs

要使用其中一个发送SMS消息 AWS SDKs，请使用与 SDK Amazon 中Publish请求对应的API操作SNSAPI。通过此请求，您可以直接向电话号码发送SMS消息。您也可使用 MessageAttributes 参数设置以下属性名称的值：

AWS.SNS.SMS.SenderID

包含 3-11 个字母数字字符或连字符 (-) 的自定义 ID，其中包含至少一个字母，并且不能包含空格。该发件人 ID 在接收设备上显示为消息发件人。例如，您可以使用自己的企业品牌让消息来源更易于识别。

对发件人的支持因国家或地区IDs而异。例如，发送至美国电话号码的消息不显示发件人 ID。有关支持发件人的国家或地区的列表IDs，请参阅Amazon SNS 支持的国家和地区。

如果您未指定发件人 ID，则在支持的国家或地区中长代码会显示为发件人 ID。对于需要按字母顺序排列的发件人 ID 的国家或地区，NOTICE则显示为发件人 ID。

此消息级别的属性会覆盖您可以通过 SetSMSAttributes 请求设置的账户级别属性 DefaultSenderID。

AWS.MM.SMS.OriginationNumber

一个由 5—14 个数字组成的自定义字符串，其中可以包括一个可选的前导加号 (+)。此数字字串在接收设备上显示为发件人的电话号码。该字符串必须与您的 AWS 账户中为目的地国家/地区配置的发件人号码相匹配。发起号码可以是 10 号码、免费DLC电话号码、 person-to-person (P2P) 长码或短码。有关更多信息，请参阅 了解 Amazon 中的原产地编号 SNS。

如果您未指定发货号，Amazon SNS 会根据您的 AWS 账户配置选择发货号。

AWS.SNS.SMS.MaxPrice

您愿意为USD发送SMS消息所花费的最高价格。如果 Amazon SNS 确定发送消息的费用将超过您的最高价格，则不会发送消息。

如果您的 month-to-date SMS成本已经超过为该属性设置的配额，则此MonthlySpendLimit属性无效。您可以使用 SetSMSAttributes 请求设置 MonthlySpendLimit 属性。

如果您要向 Amazon SNS 主题发送消息，则向订阅该主题的每个电话号码发送的每条消息均适用最高价格。

AWS.SNS.SMS.SMSType

要发送的消息类型：

- `Promotional`（默认）– 不重要的消息，例如营销消息。
- `Transactional` – 为客户事务处理提供支持的重要消息，例如多重身份验证的一次性密码。

此消息级别的属性会覆盖您可以通过 `SetSMSAttributes` 请求设置的账户级别属性 `DefaultSMSType`。

`AWS.MM.SMS.EntityId`

只有向印度的收件人发送SMS邮件时才需要此属性。

这是您的实体 ID 或委托人实体 (PE) ID，用于向印度的收件人发送SMS消息。此 ID 是一个由 1-50 个字符组成的唯一字符串，由印度电信监管局 (TRAI) 提供，用于识别您在注册的实体。TRAI

`AWS.MM.SMS.TemplateId`

只有向印度的收件人发送SMS邮件时才需要此属性。

这是您向印度收件人发送SMS消息的模板。此 ID 是一个由 1-50 个字符组成的唯一字符串，用于标识您在中注册的模板。TRAI TRAI模板 ID 必须与您为邮件指定的发件人 ID 相关联。

发送消息

以下代码示例展示了如何使用 Amazon 发布SMS消息SNS。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
```

```
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
```

```
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- 有关API详细信息，请参阅在AWS SDK for .NET API参考中发布。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
```

```
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅在AWS SDK for C++ API参考中发布。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅在AWS SDK for Java 2.x API参考中发布。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- 有关API详细信息，请参阅发布AWS SDK以获取 Kotlin API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
```

```
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅在AWS SDK for PHP API参考中发布。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- 有关API详细信息，请参阅 Python 中发布AWS SDK以供参考 (Boto3) API。

亚马逊SNSSMS活动监控

通过监控您的SMS活动，您可以跟踪目的地电话号码、成功或失败的配送、失败的原因、成本和其他信息。亚马逊通过在控制台中汇总统计数据、向亚马逊 CloudWatch发送信息以及将每日SMS使用报告发送到您指定的 Amazon S3 存储桶来提供SNS帮助。

主题

查看 Amazon SNS SMS 配送统计数据

您可以使用 Amazon SNS 控制台查看有关您最近SMS配送的统计数据。

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，将区域选择器设置为支持消息SMS传递的区域。
3. 在导航面板上，选择短信 (SMS)。
4. 在短信 (SMS) 页面的账户统计信息部分，查看您的交易和促销SMS消息发送图表。每个图表显示之前 15 天内的以下数据：
   - 传输率（成功传输的百分比）
   - 发送（传输尝试的数量）
   - 失败（传输失败的数量）

在此页面，您还可以选择 Usage（使用量）按钮转到存储您的每日使用情况报告的 Amazon S3 存储桶。有关更多信息，请参阅 订阅 Amazon SNS 每日SMS使用量报告。

使用亚马逊 CloudWatch 指标和日志监控亚马逊SNSSMS配送

您可以使用 Amazon CloudWatch 和 Amazon CloudWatch Logs 来监控您的SMS消息传送情况。

主题

查看亚马逊 CloudWatch 指标

Amazon SNS 会自动收集有关您SMS发送消息的指标，并将其推送到亚马逊 CloudWatch。您可以使用 CloudWatch 监控这些指标并创建警报，以便在指标超过阈值时提醒您。例如，您可以监控 CloudWatch 指标以了解您的SMS配送费率和 month-to-date SMS费用。

有关监控 CloudWatch 指标、设置 CloudWatch 警报和可用指标类型的信息，请参阅使用监控亚马逊 SNS话题 CloudWatch。

查看 CloudWatch 日志

您可以通过启用 Amazon 写入 Amazon SNS Logs 来收集有关成功和失败SMS消息 CloudWatch 传输的信息。对于您发送的每SMS条消息，Amazon 都会SNS写一份日志，其中包含消息价格、成功或失败状态、失败原因（如果消息失败）、消息停留时间和其他信息。

启用和查看SMS消息 CloudWatch 日志

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，将区域选择器设置为支持消息SMS传递的区域。
3. 在导航面板上，选择短信 (SMS)。
4. 在移动短信 (SMS) 页面的短信偏好设置部分，选择编辑。
5. 在下一页上，展开 Delivery status logging（传输状态日志记录）部分。
6. 对于成功采样率，请指定 Amazon SNS 将在日志中 CloudWatch 写入日志的成功SMS交付百分比。例如：

   - 要仅将失败传输写入日志，请将此值设为 0。
   - 要将 10% 的成功传输写入日志，请将其设为 10。

   如果您未指定百分比，Amazon SNS 会为所有成功交付写入日志。
7. 为提供所需的权限，请执行以下操作之一：

   - 要创建新的服务角色，请选择 Create new service role（创建新的服务角色），然后选择 Create new roles（创建新角色）。在下一页上，选择 “允许” 以授予 Amazon 对您账户资源的SNS写入权限。
   - 要使用现有的服务角色，请选择 “使用现有服务角色”，然后将ARN名称粘贴到 “成功和失败的交付IAM角色” 框中。

您指定的服务角色必须允许对账户资源进行写入访问。有关创建IAM角色的更多信息，请参阅《IAM用户指南》中的为 AWS 服务创建角色。

8. 选择 Save changes（保存更改）。
9. 返回移动短信 (SMS) 页面，转至传送状态日志部分以查看所有可用的日志。

> **Note**
>
> 根据目的地电话号码的承运人，送货日志最多可能需要 72 小时才能显示在 Amazon SNS 控制台中。

成功SMS传送的示例日志

成功交付的传SMS送状态日志将与以下示例类似：

```
{
    "notification": {
        "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
        "timestamp": "2016-06-28 00:40:34.558"
    },
    "delivery": {
        "phoneCarrier": "My Phone Carrier",
        "mnc": 270,
        "numberOfMessageParts": 1,
        "destination": "+1XXX5550100",
        "priceInUSD": 0.00645,
        "smsType": "Transactional",
        "mcc": 310,
        "providerResponse": "Message has been accepted by phone carrier",
        "dwellTimeMs": 599,
        "dwellTimeMsUntilDeviceAck": 1344
    },
    "status": "SUCCESS"
}
```

SMS传送失败的日志示例

传送失败的SMS传送状态日志将与以下示例类似：

```
{
```

```
    "notification": {
        "messageId": "1077257a-92f3-5ca3-bc97-6a915b310625",
        "timestamp": "2016-06-28 00:40:34.559"
    },
    "delivery": {
        "mnc": 0,
        "numberOfMessageParts": 1,
        "destination": "+1XXX5550100",
        "priceInUSD": 0.00645,
        "smsType": "Transactional",
        "mcc": 0,
        "providerResponse": "Unknown error attempting to reach phone",
        "dwellTimeMs": 1420,
        "dwellTimeMsUntilDeviceAck": 1692
    },
    "status": "FAILURE"
}
```

SMS配送失败原因

`providerResponse` 属性中提供失败的原因。SMS由于以下原因，邮件可能无法传送：

- 被电话运营商作为垃圾消息屏蔽
- 目的地位于阻止列表中
- 电话号码无效
- 消息正文无效
- 电话运营商已屏蔽此消息
- 电话运营商目前无法访问/不可用
- 电话已被屏蔽 SMS
- 电话位于阻止列表中
- 电话当前无法访问/可用
- 电话号码已退出
- 此传输会超过最高价格
- 尝试联系电话时发生未知错误

订阅 Amazon SNS 每日SMS使用量报告

您可以通过订阅 Amazon SNS 的每日使用报告来监控您的SMS配送。对于您每天至少发送一条SMS消息，Amazon 会将一份SNS使用情况报告作为CSV文件发送到指定的 Amazon S3 存储桶。在 S3 存储桶中显示SMS使用情况报告需要 24 小时。

主题

每日使用量报告信息

使用情况报告包含您从账户发送的每SMS封邮件的以下信息。

请注意，此报告不包含发送到已选择退出的收件人的消息。

- 消息发布时间 (inUTC)
- 消息 ID
- 目标电话号码
- 消息类型
- 传输状态
- 留言价格（英寸USD）
- 分段编号（如果一条消息过长，则会拆分为多个分段）
- 分段总数

> **Note**
>
> 如果 Amazon SNS 未收到零件号，我们会将其值设置为零。

订阅每日使用量报告

要订阅每日使用情况报告，您必须通过适当的权限创建 Amazon S3 存储桶。

为您的每日使用情况报告创建 Amazon S3 存储桶

1. 从发送SMS消息 AWS 账户 的，登录 Amazon S3 控制台。

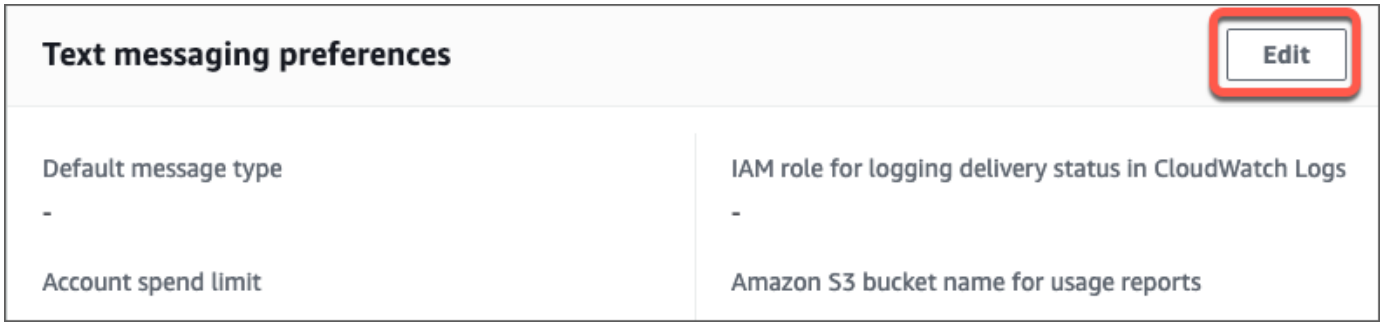
2. 选择 Create Bucket（创建存储桶）。
3. 对于存储桶名称，我们建议您输入对账户和组织唯一的名称。例如，使用模式 <my-bucket-prefix>-<account_id>-<org-id>。

   有关存储桶名称约定和限制的信息，请参阅 Amazon Simple Storage Service 用户指南中的存储桶命名规则。
4. 选择创建。
5. 在 All Buckets（所有存储桶）表中，选择存储桶。
6. 在 Permissions（权限）部分中，选择 Bucket policy（存储桶策略）。
7. 在存储桶策略编辑器窗口中，提供允许亚马逊SNS服务委托人写入您的存储桶的策略。有关示例，请参阅存储桶策略的示例。

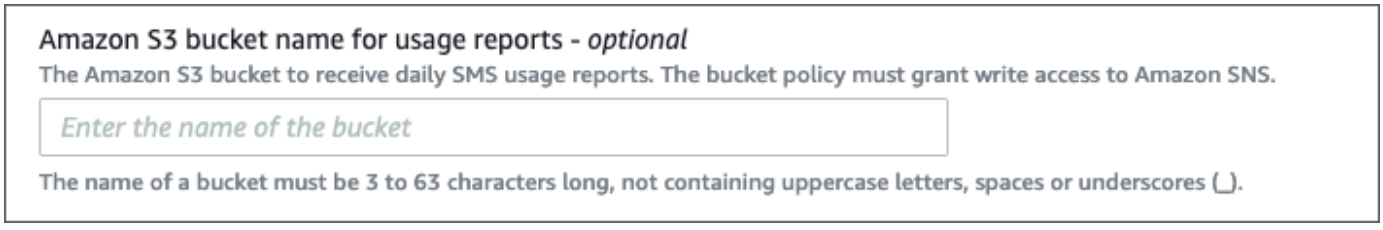
   如果您使用示例策略，请记得替换 *amzn-s3-demo-bucket* 使用您在步骤 3 中选择的存储桶名称。
8. 选择保存。

订阅每日使用情况报告

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择短信 (SMS)。
3. 在 “短信” (SMS) 页面的 “短信偏好设置” 部分，选择 “编辑”。



4. 在 Edit text messaging preferences (编辑文本消息收发首选项) 页上，在 Details (详细信息) 部分中，指定 Amazon S3 bucket name for usage reports (使用率报告的 Amazon S3 存储桶名称)。



5. 选择 Save changes（保存更改）。

存储桶策略的示例

以下策略允许 Amazon SNS 服务主体执行s3:PutObjects3:GetBucketLocation、和s3:ListBucket操作。

AWS 为所有服务提供工具，其服务委托人已被授予访问您账户中资源的权限。当 Amazon S3 存储桶策略语句中的主体为AWS 服务主体时，您可以使用 aws:SourceArn 或 aws:SourceAccount全局条件键以防止出现混淆代理人问题。要限制哪些存储桶中的哪些区域和账户可以接收每日使用情况报告，请使用 aws:SourceArn，如下面的示例所示。如果您不想限制可生成这些报告的区域，请使用 aws:SourceAccount 限制生成报告的账户。如果您不知道该ARN资源的类型，请使用aws:SourceAccount。

在创建 Amazon S3 存储桶以接收来自亚马逊的每日SMS使用报告时，使用以下示例，其中包括混乱的副手保护SNS。

```
{
 "Version": "2008-10-17",
 "Statement": [{
   "Sid": "AllowPutObject",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  },
  {
   "Sid": "AllowGetBucketLocation",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
```

```
    "Action": "s3:GetBucketLocation",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "account_id"
     },
     "ArnLike": {
      "aws:SourceArn": "arn:aws:sns:region:account_id:*"
     }
    }
   },
   {
    "Sid": "AllowListBucket",
    "Effect": "Allow",
    "Principal": {
     "Service": "sns.amazonaws.com"
    },
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "account_id"
     },
     "ArnLike": {
      "aws:SourceArn": "arn:aws:sns:region:account_id:*"
     }
    }
   }
  ]
}
```

> **Note**
>
> 您可以将使用情况报告发布到 Amazon S3 存储桶，这些存储桶由 Amazon S3 策略中 `Condition` 元素指定的 AWS 账户 拥有。要将使用情况报告发布到其他人 AWS 账户 拥有的 Amazon S3 存储桶，请参阅如何从另一个存储桶复制 S3 对象 AWS 账户？ 。

每日使用量报告的示例

在您订阅每日使用量报告后，Amazon 每天都会将包含使用情况数据的CSV文件SNS放在以下位置：

```
<amzn-s3-demo-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

每个文件可包含最多 50000 条记录。如果一天的记录超过此配额，Amazon SNS 将添加多个文件。

下面显示了一个示例报告：

```
PublishTimeUTC,MessageId,DestinationPhoneNumber,MessageType,DeliveryStatus,PriceInUSD,PartNumbe
2016-05-10T03:00:29.476Z,96a298ac-1458-4825-
a7eb-7330e0720b72,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.90084,0,1
2016-05-10T03:00:29.561Z,1e29d394-
d7f4-4dc9-996e-26412032c344,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.34322,0,1
2016-05-10T03:00:30.769Z,98ba941c-afc7-4c51-
ba2c-56c6570a6c08,1XXX5550100,Transactional,Message has been accepted by phone
 carrier,0.27815,0,1
```

管理 Amazon SNS 电话号码和订阅

Amazon SNS 提供了多种选项来管理谁从您的账户接收SMS消息。在有限的频率下，您可以选择不接收来自您帐户的SMS消息的电话号码。要停止向SMS订阅发送消息，您可以删除订阅或向其发布的主题。

主题

- 选择不接收消息 SMS
- 使用控制台管理电话号码和订阅
- 使用管理电话号码和订阅 AWS SDKs

选择不接收消息 SMS

在当地法律和法规（例如美国和加拿大）要求的情况下，SMS收件人可以通过回复邮件并提供以下任何一项来使用其设备选择退出：

- ARRET（法语）
- CANCEL
- END
- OPT-OUT
- OPTOUT

- QUIT
- REMOVE
- STOP
- TD
- UNSUBSCRIBE

要选择退出，收件人必须回复与 Amazon 发送邮件时SNS使用的相同发件号。选择退出后， AWS 账户 除非您选择电话号码，否则收件人将不再收到您发送的SMS消息。

如果电话号码订阅了某个 Amazon SNS 主题，则选择退出不会删除订阅，但除非您选择电话号码，否则SMS消息将无法发送到该订阅。

使用控制台管理电话号码和订阅

您可以使用 Amazon SNS 控制台控制哪些电话号码接收来自您账户的SMS消息。

加入已退出的电话号码

您可以查看哪些电话号码已选择不接收来自您帐户的SMS消息，也可以选择这些电话号码以继续向他们发送消息。

对于每个电话号码，您只能每隔 30 天重新加入一次。

1. 登录 Amazon SNS 控制台。
2. 在控制台菜单中，将区域选择器设置为支持消息SMS传递的区域。
3. 在导航面板上，选择短信 (SMS)。
4. 在短信 (SMS) 页面上，选择查看已退出的电话号码。Opted out phone numbers 页面将显示已退出的电话号码。
5. 选中您想要重新加入的电话号码的复选框，然后选择 Opt in。该电话号码不再被选择退出，并且会收到您向其发送的SMS消息。

删除订SMS阅

删除订SMS阅，以便在发布主题时停止向该电话号码发送SMS消息。

1. 在导航面板中，选择 Subscriptions（订阅）。
2. 选中要删除的订阅的复选框。然后选择 Actions，再选择 Delete Subscriptions。
3. 在 Delete（删除）窗口中，选择 Delete（删除）。Amazon SNS 删除订阅并显示成功消息。

删除主题

当您不想再向其订阅终端节点发布消息时，可删除主题。

1. 在导航面板上，选择 Topics（主题）。
2. 选中要删除的主题的复选框。然后选择 Actions，再选择 Delete Topics。
3. 在 Delete（删除）窗口中，选择 Delete（删除）。Amazon SNS 删除该主题并显示成功消息。

使用管理电话号码和订阅 AWS SDKs

您可以使用向 Amazon 发出编程请求，SNS并管理哪些电话号码可以从您的账户接收SMS消息。 AWS SDKs

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置AWS SDKs和凭据文件。

查看所有已退出的电话号码

要查看所有已选择退出的电话号码，请向 Amazon 提交`ListPhoneNumbersOptedOut`申请 SNSAPI。

以下代码示例演示如何使用 `ListPhoneNumbersOptedOut`。

CLI

AWS CLI

列出退出SMS留言

以下`list-phone-numbers-opted-out`示例列出了选择不接收SMS消息的电话号码。

```
aws sns list-phone-numbers-opted-out
```

输出：

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- 有关API详细信息，请参阅AWS CLI 命令参考ListPhoneNumbersOptedOut中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
```

```
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 ListPhoneNumbersOptedOut” 中的。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */
```

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 ListPhoneNumbersOptedOut” 中的。

检查电话号码是否已退出

要查看电话号码是否已选择退出，请向 Amazon 提交CheckIfPhoneNumberIsOptedOut申请SNSAPI。

以下代码示例演示如何使用 CheckIfPhoneNumberIsOptedOut。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
```

```
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use the Amazon Simple Notification Service
    /// (Amazon SNS) to check whether a phone number has been opted out.
    /// </summary>
    public class IsPhoneNumOptedOut
    {
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
```

```
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 CheckIfPhoneNumberIsOptedOut” 中的。

CLI

AWS CLI

要查看SMS留言，请选择退出电话号码

以下check-if-phone-number-is-opted-out示例检查指定的电话号码是否已选择不接收来自当前 AWS 账户的SMS消息。

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

输出：

```
{
    "isOptedOut": false
}
```

- 有关API详细信息，请参阅AWS CLI 命令参考CheckIfPhoneNumberIsOptedOut中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 CheckIfPhoneNumberIsOptedOut” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 CheckIfPhoneNumberIsOptedOut” 中的。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 CheckIfPhoneNumberIsOptedOut” 中的。

加入已退出的电话号码

要选择电话号码，请向 Amazon 提交`OptInPhoneNumber`申请SNSAPI。

对于每个电话号码，您只能每隔 30 天重新加入一次。

删除订SMS阅

要从亚马逊SNS主题中删除SMS订阅，请ARN通过向亚马逊提交`ListSubscriptions`请求来获取订阅 SNSAPI，然后将其传递ARN给`Unsubscribe`请求。

以下代码示例演示如何使用 `Unsubscribe`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过订阅取消订阅ARN某个主题。

```
    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考资料” 中的 “取消订阅”。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考资料” 中的 “取消订阅”。

CLI

AWS CLI

从主题取消订阅

以下 unsubscribe 示例将从主题删除指定的订阅。

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

此命令不生成任何输出。

- 有关API详细信息，请参阅《AWS CLI 命令参考》中的 “取消订阅”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考资料” 中的 “取消订阅”。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考资料” 中的 “取消订阅”。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- 有关API详细信息，请参阅 Kotlin API 参考中的AWS SDK取消订阅。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for PHP API参考资料” 中的 “取消订阅”。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

- 有关API详细信息，请参阅 Python (Boto3) 参考中的AWS SDK取消订阅。API

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
```

```
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅中的取消订阅AWSSDK以供SAPABAPAPI参考。

删除主题

要删除主题及其所有订阅，请向 Amazon ARN 提交`ListTopics`请求以获取该主题 SNSAPI，然后将其传递ARN给该`DeleteTopic`请求。

以下代码示例演示如何使用 `DeleteTopic`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

按主题删除主题ARN。

```
    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
```

```
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 DeleteTopic” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 DeleteTopic” 中的。

CLI

AWS CLI

删除SNS主题

以下delete-topic示例删除了指定的SNS主题。

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

此命令不生成任何输出。

- 有关API详细信息，请参阅AWS CLI 命令参考DeleteTopic中的。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}
```

```
// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- 有关API详细信息，请参阅 “AWS SDK for Go API参考 DeleteTopic” 中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 DeleteTopic” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
```

```
  // }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 DeleteTopic” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- 有关API详细信息，请参阅DeleteTopic中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 DeleteTopic” 中的。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- 有关API详细信息，请参阅DeleteTopic中的 AWS SDKPython (Boto3) API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
TRY.
    lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
    MESSAGE 'SNS topic deleted.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关API详细信息，请参阅DeleteTopic中的AWS SDK以供SAPABAPAPI参考。

Amazon SNS 支持的国家和地区

> **Important**
>
> 自 2023 年 8 月 31 日起，向美国及其领土（关岛、波多黎各、美属萨摩亚群岛和US维尔京群岛）发送SMS消息时需要使用诸如亚马逊 SNS 10 DLC号码或免费电话号码之类的专用号码。

目前，Amazon SNS 支持在以下 AWS 区域SMS发送消息：

区域名称	区域	端点	协议
美国东部（俄亥俄州）	us-east-2	sns.us-east-2.amazonaws.com	HTTP和 HTTPS
美国东部（弗吉尼亚州北部）	us-east-1	sns.us-east-1.amazonaws.com	HTTP和 HTTPS

区域名称	区域	端点	协议
美国西部（加利福尼亚北部）	us-west-1	sns.us-west-1.amazonaws.com	HTTP和 HTTPS
美国西部（俄勒冈州）	us-west-2	sns.us-west-2.amazonaws.com	HTTP和 HTTPS
非洲（开普敦）	af-south-1	sns.af-south-1.amazonaws.com	HTTP和 HTTPS
亚太地区（海得拉巴）	ap-south-2	sns.ap-south-2.amazonaws.com	HTTP和 HTTPS
亚太地区（雅加达）	ap-southeast-3	sns.ap-southeast-3.amazonaws.com	HTTP和 HTTPS
亚太地区（墨尔本）	ap-southeast-4	sns.ap-southeast-4.amazonaws.com	HTTP和 HTTPS
亚太地区（孟买）	ap-south-1	sns.ap-south-1.amazonaws.com	HTTP和 HTTPS
Asia Pacific (Osaka)	ap-northeast-3	sns.ap-northeast-3.amazonaws.com	HTTP和 HTTPS
亚太地区（新加坡）	ap-southeast-1	sns.ap-southeast-1.amazonaws.com	HTTP和 HTTPS
亚太地区（悉尼）	ap-southeast-2	sns.ap-southeast-2.amazonaws.com	HTTP和 HTTPS
Asia Pacific (Tokyo)	ap-northeast-1	sns.ap-northeast-1.amazonaws.com	HTTP和 HTTPS
加拿大（中部）	ca-central-1	sns.ca-central-1.amazonaws.com	HTTP和 HTTPS
加拿大西部（卡尔加里）	ca-west-1	sns.ca-west-1.amazonaws.com	HTTP和 HTTPS

区域名称	区域	端点	协议
欧洲（法兰克福）	eu-central-1	sns.eu-central-1.amazonaws.com	HTTP和 HTTPS
欧洲地区（爱尔兰）	eu-west-1	sns.eu-west-1.amazonaws.com	HTTP和 HTTPS
欧洲（伦敦）	eu-west-2	sns.eu-west-2.amazonaws.com	HTTP和 HTTPS
欧洲（米兰）	eu-south-1	sns.eu-south-1.amazonaws.com	HTTP和 HTTPS
欧洲（巴黎）	eu-west-3	sns.eu-west-3.amazonaws.com	HTTP和 HTTPS
欧洲（西班牙）	eu-south-2	sns.eu-south-2.amazonaws.com	HTTP和 HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	sns.eu-north-1.amazonaws.com	HTTP和 HTTPS
欧洲（苏黎世）	eu-central-2	sns.eu-central-2.amazonaws.com	HTTP和 HTTPS
以色列（特拉维夫）	il-central-1	sns.il-central-1.amazonaws.com	HTTP和 HTTPS
中东（巴林）	me-south-1	sns.me-south-1.amazonaws.com	HTTP和 HTTPS
中东 (UAE)	me-central-1	sns.me-central-1.amazonaws.com	HTTP和 HTTPS
南美洲（圣保罗）	sa-east-1	sns.sa-east-1.amazonaws.com	HTTP和 HTTPS
AWS GovCloud （美国东部）	us-gov-east-1	sns。 us-gov-east-1.amazonaws.com	HTTP和 HTTPS

区域名称	区域	端点	协议
AWS GovCloud （美国西部）	us-gov-west-1	sns。 us-gov-west-1.amazonaws.com	HTTP和 HTTPS

您可以使用 Amazon SNS 向以下国家和地区发送SMS消息：

> **Note**
>
> IDs在支持的国家/地区使用发件人可以改善SMS配送。

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
A						
阿富汗	AF	93	否	否	是	不支持
阿尔巴尼亚	AL	355	否	否	是	不支持
阿尔及利亚	DZ	213	否	否	是	不支持
安道尔	AD	376	否	否	是	不支持
安圭拉岛	AI	1-264	否	否	是	不支持
安提瓜和巴布达	AG	1-268	否	否	是	不支持
阿根廷	AR	54	是	否	否	否
亚美尼亚	AM	374	否	否	是	不支持
阿鲁巴岛	AW	297	否	否	是	不支持
澳大利亚	AU	61	否	是	需要注册[1]	是
奥地利	AT	43	是	是	是	是

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
阿塞拜疆	AZ	994	否	否	是	不支持
B						
巴哈马	BS	1-242	否	否	否	否
巴林	BH	973	否	否	是	不支持
孟加拉国	BD	880	否	否	是	不支持
巴巴多斯	BB	1-246	否	否	是	不支持
白俄罗斯	BY	375	否	否	需要注册[1]	否
比利时	BE	32	否	是	否	是
伯利兹	BZ	501	否	否	是	不支持
百慕大	BM	1-441	否	否	是	不支持
不丹	BT	975	否	否	是	不支持
玻利维亚	BO	591	否	否	是	不支持
波斯尼亚和黑塞哥维那	BA	387	否	否	是	不支持
博茨瓦纳	BW	267	否	否	是	不支持
巴西	BR	55	是	否	否	是
文莱	BN	673	否	否	是	不支持
保加利亚	BG	359	是	否	是	是
布基纳法索	BF	226	否	否	是	不支持
布隆迪	BL	257	否	否	是	不支持

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
C						
柬埔寨	KH	855	否	否	是	不支持
喀麦隆	CM	237	否	否	是	不支持
加拿大	CA	1	是	是	否	是
佛得角	CV	238	否	否	是	不支持
开曼群岛	KY	1-345	否	否	否	否
中非共和国	CF	236	否	否	是	不支持
乍得	TD	235	否	否	是	不支持
智利	CL	56	是	是	否	是
中国	CN	86	是	不支持	否[2]	是
哥伦比亚	CO	57	是	是	否	是
科摩罗	KM	269	否	否	是	不支持
库克群岛	CK	682	否	否	是	是
哥斯达黎加	CR	506	否	否	否	否
克罗地亚	HR	385	否	否	是	不支持
塞浦路斯	CY	357	否	否	是	不支持
捷克（捷克共和国）	CZ	420	否	是	否	是
D						

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
刚果民主共和国	CD	243	否	否	是	不支持
丹麦	DK	45	是	是	是	是
吉布提	DJ	253	否	否	是	不支持
多米尼加	DM	1-767	否	否	是	不支持
多米尼加共和国	DO	1-809、1-829、1-849	是	否	否	是
E						
厄瓜多尔	EC	593	是	否	否	是
埃及	EG	20	是	不支持	需要注册[1]	是
萨尔瓦多	SV	503	否	否	否	否
赤道几内亚	GQ	240	否	否	是	不支持
厄立特里亚	ER	291	否	否	是	不支持
爱沙尼亚	EE	372	否	是	是	是
埃塞俄比亚	ET	251	否	否	是	不支持
F						
法罗群岛	FO	298	否	否	是	不支持
斐济	FJ	679	否	否	是	不支持
芬兰	FI	358	是	是	是	是
法国	FR	33	是	否	是	是

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
法属圭亚那	GF	594	否	否	是	不支持
法属玻里尼西亚	PF	689	否	否	是	不支持
G						
加蓬	GA	241	否	否	是	不支持
冈比亚	GM	220	否	否	是	不支持
格鲁吉亚	GE	995	否	否	是	不支持
德国	DE	49	是	是	是	是
加纳	GH	233	否	否	是	不支持
直布罗陀	GI	350	否	否	是	不支持
希腊	GR	30	否	否	是	不支持
格陵兰	GL	299	否	否	是	不支持
格林纳达	GD	1-473	否	否	是	不支持
瓜德罗普	GP	590	否	否	是	不支持
关岛	GU	1-671	否	否	否	是
危地马拉	GT	502	否	否	否	否
根西岛	GG	44-1481	否	否	是	不支持
几内亚	GN	224	否	否	是	不支持
几内亚比绍	GW	245	否	否	是	不适用
圭亚那	GY	592	否	否	是	不支持

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人 IDs	支持双向 SMS
H						
海地	H	509	否	否	是	不支持
洪都拉斯	HN	504	否	否	是	不支持
中国香港	HK	852	否	是	是	是
匈牙利	HU	36	否	是	否	是
I						
冰岛	IS	354	否	否	是	不支持
印度	IN	91	是	是[4]	需要注册[3]	是
印度尼西亚	ID	62	否	否	是	不支持
伊拉克	IQ	964	否	否	是	不支持
爱尔兰	IE	353	否	是	是	是
马恩岛	IM	44-1624	否	否	是	不支持
以色列	IL	972	否	是	是	是
意大利	IT	39	是	是	是	是
科特迪瓦	CI	225	否	否	是	不支持
J						
牙买加	JM	1-876	否	否	是	不支持
日本	JP	81	是	是	是	是
泽西岛	JE	44-1534	否	是	是	是

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
约旦	JO	962	否	否	需要注册[1]	否
K						
哈萨克	KZ	7	否	否	需要注册[1]	否
肯尼亚	KE	254	否	否	是	不支持
科索沃	XK	383	否	否	是	不支持
科威特	KW	965	否	否	需要注册[1]	否
吉尔吉斯斯坦	KG	996	否	否	是	不支持
L						
老挝	LA	856	否	否	是	不支持
拉脱维亚	LV	371	否	否	是	不支持
黎巴嫩	LB	961	否	否	是	不支持
莱索托	LS	266	否	否	是	不支持
利比里亚	LR	231	否	是	不支持	
利比亚	LY	218	否	否	是	不支持
列支敦士登	LI	423	否	否	是	不支持
立陶宛	LT	370	否	是	是	是
卢森堡	LU	352	否	是	是	是
M						
澳门	MO	853	否	否	是	不支持

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
马其顿	MK	389	否	否	是	不支持
马达加斯加	MG	261	否	否	是	不支持
马拉维	MW	265	否	否	是	不支持
马来西亚	MY	60	是	否	否	是
马尔代夫	MV	960	否	否	是	不支持
Mali	ML	223	否	否	是	不支持
马耳他	MT	356	否	否	是	不支持
马绍尔群岛	MH	692	否	否	否	否
马提尼克	MQ	596	否	否	是	不支持
毛里塔尼亚	MR	222	否	否	是	不支持
毛里求斯	MU	230	否	否	是	不支持
马约特岛	YT	262	否	否	是	不支持
墨西哥	MX	52	是	否	否	是
密克罗尼西亚联邦	FM	691	否	否	否	否
摩尔多瓦	MD	373	否	否	是	不支持
摩纳哥	MC	377	否	否	否	否
蒙古	MN	976	否	否	是	不支持
黑山共和国	ME	382	否	否	是	不支持
蒙特塞拉特岛	MS	1-664	否	否	是	不支持

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
摩洛哥	MA	212	是	否	是	是
莫桑比克	MZ	258	否	否	否	否
缅甸	MM	95	否	是	是	是
否						
纳米比亚	NA	264	否	否	是	不支持
尼泊尔	NP	977	否	否	是	不支持
荷兰	NL	31	是	是	是	是
荷属安的列斯	AN	599	否	否	是	不支持
新喀里多尼亚	NC	687	否	否	是	不支持
新西兰 [6]	NZ	64	是	否	否	是
尼加拉瓜	NI	505	否	否	否	否
尼日尔	NE	227	否	否	是	不支持
尼日利亚	NG	234	否	否	是	不支持
纽埃岛	NU	683	否	否	是	不支持
挪威	NO	47	否	是	是	是
O						
阿曼	OM	968	否	否	否	不适用
P						

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
巴基斯坦	PK	92	否	否	是	不适用
巴勒斯坦	PS	970	否	否	是	不支持
巴拿马	PA	507	否	否	是	不支持
巴布亚新几内亚	PG	675	否	否	是	不支持
巴拉圭	PY	595	否	否	否	否
秘鲁	PE	51	是	否	否	是
菲律宾	PH	63	否	是	需要注册[1]	是
波兰	PL	48	否	是	是	是
葡萄牙	PT	351	否	是	是	是
波多黎各	PR	1-797、1-939	否	否	否	是
Q						
卡塔尔	QA	974	否	否	需要注册[1]	否
R						
刚果共和国	CG	242	否	否	否	否
留尼旺岛（法国）	RE	262	否	否	是	不支持
罗马尼亚	RO	40	否	是	是	是
俄罗斯	RU	7	是	不支持	需要注册[1]	是
卢旺达	RW	250	否	否	是	不支持

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
S						
圣基茨和尼维斯	KN	1-869	否	否	否	否
圣卢西亚岛	LC	1-758	否	否	否	否
萨摩亚群岛	WS	685	否	否	是	不支持
圣马力诺	SM	378	否	否	是	不支持
圣多美和普林西比	ST	239	否	否	是	不支持
沙特阿拉伯	SA	966	否	是[4]	需要注册[1]	否
塞内加尔	SN	221	否	否	是	不支持
塞尔维亚	RS	381	否	否	是	不支持
塞舌尔	SC	248	否	否	是	不支持
塞拉利昂	SL	232	否	否	是	不支持
新加坡	SG	65	是	是	是[5]	是
斯洛伐克	SK	421	否	是	是	不支持
斯洛文尼亚	SI	386	否	否	是	不支持
所罗门群岛	SB	677	否	否	是	不支持
索马里	SO	252	否	否	是	不支持
南非	ZA	27	是	是	否	是
韩国	KR	82	否	否	否	否

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
南苏丹	SS	211	否	否	是	不支持
西班牙	ES	34	是	是	是	是
斯里兰卡	LK	94	否	否	需要注册[1]	否
苏里南	SR	597	否	否	是	不支持
斯威士兰	SZ	268	否	否	是	不支持
瑞典	SE	46	是	是	是	是
瑞士	CH	41	否	是	是	是
T						
中国台湾	TW	886	否	是	否	是
塔吉克斯坦	TJ	992	否	否	是	不支持
坦桑尼亚	TX	255	否	否	是	不支持
泰国	TH	66	否	是	需要注册[1]	是
东帝汶	TL	670	否	否	是	不支持
多哥	TG	228	否	否	是	不支持
汤加	TO	676	否	否	是	不支持
特立尼达和多巴哥	TT	1-868	否	否	是	不支持
突尼斯	TN	216	否	否	是	不支持
土耳其	TR	90	否	否	需要注册[1]	否
土库曼斯坦	TM	993	否	否	否	否

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人IDs	支持双向SMS
特克斯和凯科斯群岛	TC	1-649	否	否	是	不支持
图瓦卢	TC	688	否	否	是	不支持
U						
乌干达	UG	256	否	否	是	不支持
乌克兰	UA	380	否	是	是	是
阿拉伯联合酋长国(UAE)	AE	971	是	是	需要注册[1]	是
英国	GB	44	是	是	是	是
美国	US	1	是	是	否	是
乌拉圭	UY	598	是	否	否	是
乌兹别克斯坦	UZ	998	否	否	是	不支持
V						
瓦努阿图	VU	678	否	否	是	不支持
委内瑞拉	VE	58	否	否	否	否
越南	VN	84	否	否	需要注册[1]	否
英属维尔京群岛	VG	1-284	否	否	是	不支持
美属维尔京群岛	VI	1-340	否	否	否	是

国家或地区	ISO代码	拨号代码	支持短代码	支持长代码	支持发件人 IDs	支持双向 SMS
W						
X						
Y						
也门	YE	967	否	否	是	不支持
Z						
赞比亚	ZM	260	否	否	是	不支持
津巴布韦	ZW	263	否	否	是	不支持

注意

1. 发送人需要使用预先注册的发送人 ID（由字母组成）。要向其请求发件人 ID AWS Support，请参阅请求发件人IDs向 Amazon 发送SMS消息 SNS。一些国家/地区要求发送人符合特定要求或者遵守特定限制才能获得批准。在这些情况下， AWS Support 可能会在您提交发件人身份请求后与您联系以获取更多信息。
2. 发件人需要对计划发送的每种类型的消息使用预先注册的模板。如果发件人不符合此要求，他们的消息将被阻止。要注册模板，请使用打开 Amaz SNS SMS on 问题 AWS Support。创建案例时，请提供用于请求发件人 ID 的相同信息。有关更多信息，请参阅 请求发件人IDs向 Amazon 发送SMS消息 SNS。一些国家/地区需要发件人符合额外的特定要求或者遵守特定限制才能获取审批。在这些情况下， AWS Support 可能会要求您提供更多信息。

   > **Note**
   >
   > 要向中国发送消息，您必须先注册模板 AWS Support 以获得批准。

3. 发送人需要使用预先注册的发送人 ID（由字母组成）。需要额外注册步骤。有关更多信息，请参阅印度的 Amazon SNS 发件人 ID 注册要求。
4.

这些国家/地区的长代码仅支持入站消息收发。换句话说，您不能使用这些长代码向 接收人发送消息，但可以使用它们从 接收人接收消息。如果您使用字母发件人 ID 发送邮件，这些长代码是允许收件人选择退出的有用方法，因为发件人IDs仅支持出站邮件。

5. Amazon SNS 可以使用已在新加坡发件人 ID 注册表 (SSIR) 中注册的SMS发件人 ID 向新加坡发送SMS流量，该注册表由新加坡信息通信媒体发展局 (IMDA) 创建。有关使用新加坡发件人 ID 的要求的更多信息，请参阅新加坡的 Amazon SNS 发件人 ID 注册要求。

   您还可以使用未注册的发件人IDs或其他来源身份类型（例如短代码或长码）在新加坡发送SMS流量。

6. 如果没有专用的短代码，Amazon SNS 仍会尝试使用共享的短代码池向新西兰收件人发送消息。由于当地运营商对共享号码的限制，通过这些共享号码发送时的送达率建立在尽最大努力的基础之上。因此，Amazon SNS 强烈建议为发送到新西兰的所有流量购买专用的短代码。包含的消息URLs必须通过专门的短代码流程列入允许名单。有关购买短代码的更多信息，请参阅向 Amazon 申请SMS发送消息的专用短代码 SNS。

亚马逊SNSSMS最佳实践

手机用户对未经请求SMS的消息的容忍度往往非常低。不请自来的SMS活动的回复率几乎总是很低，因此您的投资回报率会很差。

此外，移动电话运营商会持续对批量SMS发送者进行审计。他们会从其确定发送未经请求的消息的数量中限制或阻止消息。

发送未经请求的内容也是一种违反 AWS 可接受使用策略的行为。Amazon SNS 团队会定期对SMS活动进行审计，如果发现您发送的是未经请求的消息，则可能会限制或阻止您发送消息。

最后，在许多国家、地区和司法管辖区，发送未经请求SMS的消息会受到严厉的处罚。例如，在美国，《电话消费者保护法》(TCPA) 规定，消费者收到的每封未经请求的消息都有权获得500至1,500美元的赔偿金（由发件人支付）。

本部分介绍了几个可帮助您提升客户参与度并避免代价高昂的处罚的最佳实践。但请注意，本节不包含法律建议。务必咨询律师来获取法律建议。

主题

- 遵守法律、法规和运营商要求

- 获取权限
- 不要发送到旧名单
- 审核客户列表
- 保留记录
- 提供清晰、诚实、简洁的信息
- 适当地响应
- 基于参与度调整您的发送
- 在适当时间发送
- 避免跨通道疲劳
- 使用专用短代码
- 验证您的目标电话号码
- 设计时要考虑冗余
- SMS限制和限制
- 管理退出关键词
- CreatePool
- PutKeyword
- 管理号码设置
- SMSAmazon 中的字符限制 SNS

遵守法律、法规和运营商要求

如果您违反客户所在地的法律和法规，您可能面对重大罚款和处罚。因此，了解您开展业务的每个国家或地区与SMS消息传递相关的法律至关重要。

以下列表包含适用于全球主要市场SMS通信的主要法律的链接。

- 美国：1991年的《电话消费者保护法》（也称为TCPA）适用于某些类型的SMS消息。有关更多信息，请访问美国联邦通信委员会 (Federal Communications Commission) 网站上的规则和法规。
- 英国：2003 年《隐私和电子通信（欧共体指令）条例》（也称为PECR）适用于某些类型的SMS消息。有关更多信息，请参阅什么是PECR？ 在英国信息专员办公室的网站上。
- 欧盟：《2002年隐私和电子通信指令》（有时也称为 ePrivacy 指令）适用于某些类型的SMS消息。有关更多信息，请访问 Europa.eu 网站，查看该法律的完整文本。

- 加拿大：《打击互联网和无线垃圾邮件法》，通常被称为加拿大的反垃圾邮件法CASL，或适用于某些类型的SMS邮件。有关更多信息，请访问加拿大国会 (Parliament of Canada) 网站，查看该法律的完整文本。
- 日本：《特定电子邮件传输管理法》可能适用于某些类型的SMS消息。有关更多信息，请访问日本内务与通信省 (Japanese Ministry of Internal Affairs and Communications) 网站，查看日本的垃圾邮件应对措施。

作为发件人，即使您的公司或部门并非位于这些国家/地区之一，这些法律也可能适用于您。此列表中的一些法律最初是为处理不请自来的电子邮件或电话而制定的，但已被解释或扩展为适用于SMS消息。其他国家和地区可能有自己的与SMS消息传输相关的法律。请咨询您的客户所在的每个国家/地区的律师以获得法律建议。

在许多国家/地区，本地运营商具有确定哪些类型的流量可通过其网络传输的最终权力。这意味着运营商可能会对超出当地法律最低要求的SMS内容施加限制。

获取权限

在您计划发送特定类型的消息时，切勿向未明确要求接收此类消息的收件人发送消息。不要共享选择加入名单，即使在同一家公司内的部门之间也是如此。

如果收件人可以使用在线表格进行注册以接收您的消息，请添加预防系统，防止自动化脚本在人员不知道的情况下进行订阅。您还应该限制用户在单个会话中可以提交某个电话号码的次数。

当您收到SMS选择加入请求时，请向收件人发送一条消息，要求他们确认是否要接收您的消息。请勿在收件人确认订阅之前向其发送任何其他消息。订阅确认消息可能类似于以下示例：

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may
apply. Reply HELP for help, STOP to cancel.
```

维护包含每个选择加入请求和确认的日期、时间和来源的记录。这在运营商或监管机构请求它的情况下可能会有用，并且还可以帮助您执行客户列表的例行审核。

选择加入工作流程

在某些情况下（例如美国免费电话或短代码注册），移动运营商要求您提供整个选择加入工作流程的模型或屏幕截图。模型或屏幕截图必须与收件人将要完成的选择加入工作流程非常接近。

您的模型或屏幕截图应包括下面列出的所有必要披露信息，以保持最高的合规水平。

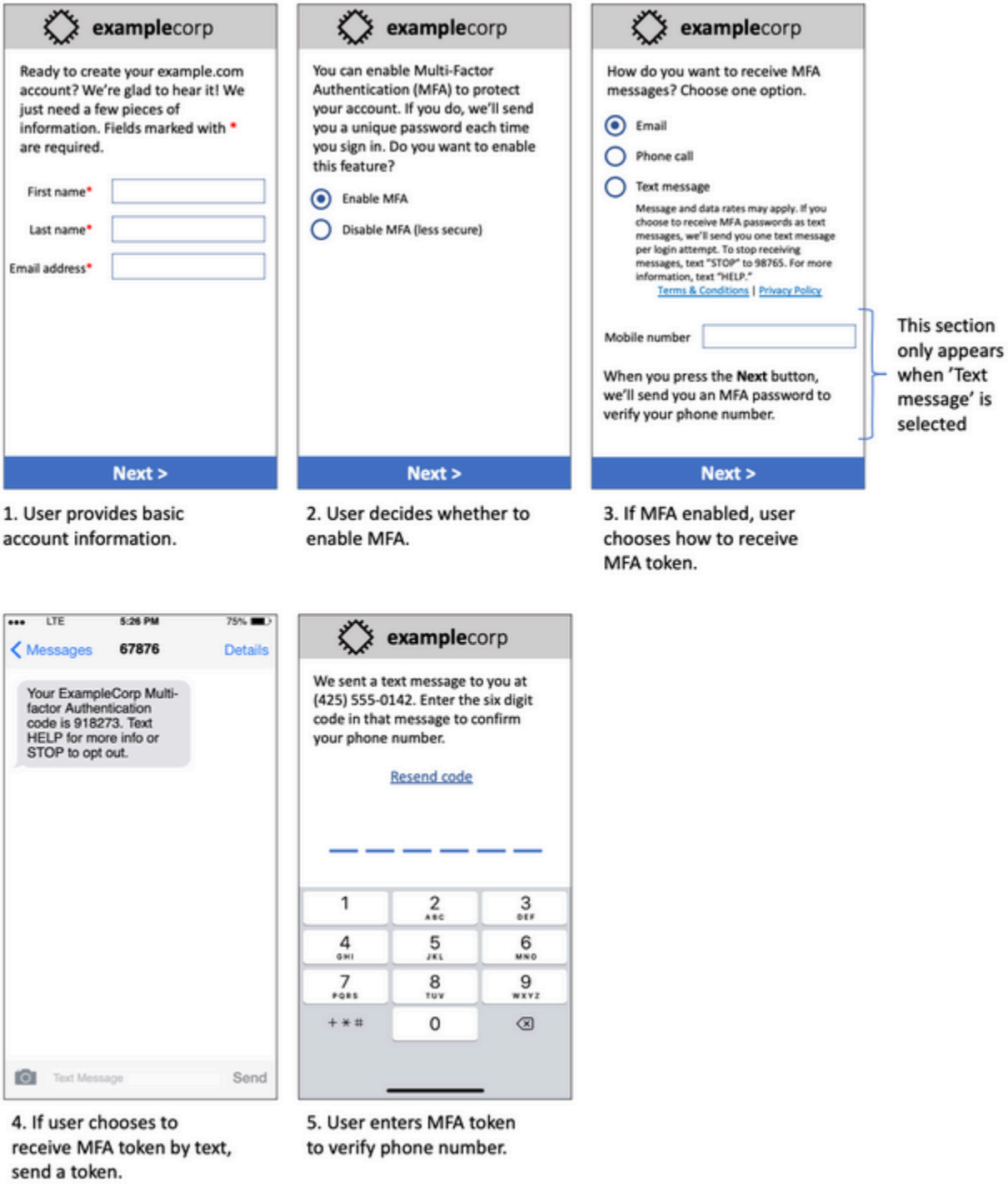
必要的披露信息

- 对您将通过程序发送的消息使用场景的描述。
- 陈述“可能会收取消息和数据费用”。
- 说明收件人收到您的消息的频率。例如，定期消息发送程序可以说明“每周一条消息”。一次性密码或多重验证使用场景可以说明“消息频率会有变化”或“每次登录尝试一条消息”。
- 指向您的条款和条件以及隐私策略文档的链接。

不合规的选择加入方法的常见拒绝原因

- 提供的公司名称与模型或屏幕截图中提供的名称不相符。在选择加入工作流程描述中需要解释任何不明确的关系。
- 似乎会向收件人发送消息，但事先并未征得明确同意。所有消息发送都必须征得明确同意。
- 似乎需要接收短信才能注册服务。如果工作流程没有提供其他形式（例如，电子邮件或语音呼叫）作为选择加入消息的替代方法，则这不合规。
- 关于选择加入的说明完全包含在服务条款中。披露内容应始终在选择加入时提交给接收人，而不是包含在关联的策略文档中。
- 客户同意接收您发送的一种类型的消息，而您向他们发送了其他类型的文本消息。例如，他们同意接收一次性密码，但还会向其发送投票和调查消息。
- 没有向收件人提供所需的披露信息（如上所列）。

以下示例符合移动运营商对多重验证使用场景的要求。



它包含最终的文本和图像，展示了完整的选择加入流程，并附有注释。在选择加入流程中，客户必须采取意图明确的操作来表示同意接收文本消息，且流程中包含所有必需的披露信息。

其他选择加入工作流程类型

在符合此处所述要求的前提下，移动运营商还会接受应用程序和网站之外的选择加入工作流程，例如口头或书面选择加入。合规的选择加入工作流程以及口头或书面脚本需要征求收件人的明确同意，允许接收特定的消息类型。这方面的例子包括客户支持座席使用的语音脚本，用来在记录到服务数据库之前征得同意，或者在宣传单上列出的电话号码。要提供这些选择加入工作流程类型的模型，您可以提供选择加入脚本、营销材料或收集号码的数据库的屏幕截图。如果选择加入的场景不明确或使用场景超过一定数量，则移动运营商可能会对这些使用场景有其他疑问。

不要发送到旧名单

人们经常更换电话号码。两年前已获取联系同意书的电话号码今天可能是其他人在使用。不要在新的消息发送程序中使用旧的电话号码列表；如果这样做，则可能会因为号码不再使用而导致一些消息失败，而另外一些人可能会因为不记得一开始同意过会选择退出。

审核客户列表

如果您定期发送SMS营销活动，请定期审核您的客户列表。审核您的客户列表可确保仅接收您的消息的客户是有兴趣接收这些消息的人员。

审核您的列表时，向每个选择加入的客户发送提醒他们已订阅的消息，并为他们提供有关取消订阅的信息。提醒消息可能类似于以下示例：

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

保留记录

保留记录，显示每位客户何时请求接收您的SMS消息，以及您向每位客户发送了哪些消息。世界上许多国家和地区都要求SMS发件人以易于检索的方式保存这些记录。移动运营商还可能会随时向您请求提供此信息。您所必须提供的确切信息因国家或地区而异。有关记录保存要求的更多信息，请查看客户所在的每个国家或地区的商业SMS消息传送法规。

有时，运营商或监管机构会要求我们提供客户选择接收您的消息的证据。在这种情况下，请与您 AWS Support 联系并提供承运人或代理机构所需的信息清单。如果您无法提供必要的信息，我们可能会暂停您发送其他SMS消息的权限。

提供清晰、诚实、简洁的信息

SMS是一种独特的媒介。160-的character-per-message 限制意味着您的消息必须简洁。你可能在其他通信渠道（例如电子邮件）中使用的技巧可能不适用于该SMS渠道，在处理消息时甚至可能显得不诚

实或具有欺骗性。SMS如果消息中的内容与最佳实践不一致，收件人可能会忽略您的消息；在最糟糕的情况下，移动运营商可能会将您的消息标识为垃圾消息，以后会屏蔽来自您电话号码的消息。

本节提供了一些创建有效SMS消息正文的提示和想法。

将自己标识为发件人

收件人应该能够立即分辨出是您发布的消息。遵循此最佳实践的发件人会在每条消息的开头添加一个识别名称 (“计划名称”) 。

请勿执行以下操作：

```
Your account has been accessed from a new device. Reply Y to confirm.
```

试试这个：

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

不要试图让你的消息看起来像一条 person-to-person 消息

一些营销人员倾向于让他们的SMS信息看起来像来自个人，从而为他们的信息增添个人风格。但是，这种技巧可能会让您的消息看起来像是网络钓鱼尝试。

请勿执行以下操作：

```
Hi, this is Jane. Did you know that you can save up to 50% at
Example.com? Click here for more info: https://www.example.com.
```

试试这个：

```
ExampleCorp Offers: Save 25-50% on sale items at Example.com. Click here
to browse the sale: https://www.example.com. Text STOP to opt-out.
```

在涉及到金钱时请谨慎

诈骗者经常会利用人们省钱和获利的渴望。不要提供好得难以置信的优惠。不要利用金钱的诱惑来骗人。不要使用货币符号来指示金钱。

请勿执行以下操作：

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

试试这个：

```
ExampleCorp Offers: Your ExampleCorp insurance policy gets you discounts
at 2300+ repair shops nationwide. More info at https://www.example.com.
Text STOP to opt-out.
```

仅使用必要的字符

品牌通常倾向于在消息中使用 ™ 或 ® 等商标符号来保护自己的商标。但是，这些符号不是可以包含在 160 个字符的消息中的标准字符SMS集（称为字GSM母）的一部分。当您发送某条包含这些字符的消息时，会使用不同的字符编码系统自动发送您的消息，而这一系统的每段消息仅支持 70 个字符。因此，您的消息可能会分为几段。由于您需要为发送的每段消息付费，因此发送整条消息的费用可能会超出您的预期。此外，收件人可能会收到您发来的多条连续消息，而不是一条消息。有关SMS字符编码的更多信息，请参见SMSAmazon 中的字符限制 SNS。

请勿执行以下操作：

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

试试这个：

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> 前面的两个示例几乎相同，但第一个示例包含注册商标符号 (®)，该符号不是GSM字母表的一部分。因此，第一个示例作为两段消息发送，而第二个示例作为一段消息发送。

使用有效且安全的链接

如果您的消息包含链接，请仔细检查链接以确保链接可以正常工作。在公司网络之外的设备上测试您的链接，确保能够正确解析链接。由于消息的长度限制为 160 个字符，因此URLs可以将很长的SMS消息分成多条消息。您应该使用重定向域名来提供简短的域名URLs。但是，您不应使用免费的链接缩短服务（如 tinyurl.com 或 bitly.com），因为运营商倾向于过滤掉包含这些域中链接的消息。不过，只要链接指向专供贵公司或机构使用的域，您就可以使用付费的链接缩短服务。

请勿执行以下操作：

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

试试这个：

```
ExampleCorp Offers: Today only, get an exclusive deal on an ExampleCorp
Widget. See https://a.co/cFKmaRG for more info. Text STOP to opt-out.
```

限制使用的缩略语数量

该SMS频道的160个字符限制使一些发件人认为他们需要在邮件中广泛使用缩写。但是，对许多读者来说，过度使用缩写会显得不专业，并可能导致一些用户将您的消息举报为垃圾消息。您完全可以编写流畅的消息而不过多地使用缩写。

请勿执行以下操作：

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

试试这个：

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

适当地响应

当收件人回复您的消息时，请确保您使用有用的信息进行响应。例如，当客户使用关键字 HELP “” 回复您的一条消息时，向他们发送有关他们订阅的计划、您每月要发送的消息数量以及他们与您联系以获取更多信息的方式的信息。HELP响应可能类似于以下示例：

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

当客户回复关键字 STOP “” 时，请告知他们不会再收到任何消息。STOP响应可能类似于以下示例：

```
You're unsubscribed from ExampleCorp alerts. No more messages will be sent.
Reply HELP, email help@example.com, or call 425-555-0199 for more info.
```

基于参与度调整您的发送

您客户的优先级可能随着时间推移而发生变化。如果客户发现您的消息不再有用，则他们可能会选择完全不再使用您的消息，或者甚至将您的消息报告为未经请求的消息。出于这些原因，您必须基于客户参与度调整您的发送活动。

对于与您的消息互动很少的客户，您应调整相应的消息发送频率。例如，如果向参与的客户每周发送消息，您可以为参与度较低的客户创建单独的每月摘要文件。

最后，从您的客户列表中删除完全未参与的客户。此步骤可防止客户对您的消息感到沮丧。这还可为您节省资金并且帮助保护您作为发件人的声誉。

在适当时间发送

仅在正常白天工作时间内发送消息。如果您在晚餐时间或午夜发送消息，则很可能您的客户将从您的列表中取消订阅以避免被打扰。此外，当您的客户无法立即回复SMS消息时，发送消息是没有意义的。

如果您将活动或旅程发送给非常多的受众，请仔细检查您的发送号码的吞吐速率。将收件人数量除以您的吞吐速率，确定向所有收件人发送消息需要多长时间。

避免跨通道疲劳

在广告活动中，如果您使用多个沟通渠道（例如电子邮件和推送消息），请不要在每个渠道中发送相同的消息。SMS当时在多个渠道中同时发送相同消息时，您的客户可能会认为您的发送行为很烦人而不是有用。

使用专用短代码

如果使用短代码，请为每个品牌和每种类型的消息维护单独的短代码。例如，如果您的公司有两个品牌，请为每个品牌使用单独的短代码。同样，如果您发送事务和促销消息，请为每种类型的消息使用单独的短代码。要了解有关请求短代码的更多信息，请参阅向 Amazon 申请SMS发送消息的专用短代码SNS。

验证您的目标电话号码

当您通过 Amazon 发送SMS消息时SNS，您需要为发送的每个消息部分付费。您为每段消息支付的价格因收件人所在的国家或地区而异。有关SMS定价的更多信息，请参阅 Amazon SNS 定价。

当 Amazon SNS 接受发送SMS消息的请求时（致电给我们 SendMessagesAPI，或者由于活动或旅程的启动），则您需要支付发送该消息的费用。即使预期的收件人实际上没有收到消息，您也需要支付费

用。例如，如果收件人的电话号码已停用，或者您向其发送消息的号码不是有效的手机号码，但仍会向您收取发送消息的费用。

Amazon SNS 接受有效的SMS消息发送请求并尝试传送这些请求。因此，您应该验证向其发送消息的电话号码是否为有效的手机号码。您可以使用 Amazon SNS 电话号码验证服务来确定电话号码是否有效以及该号码的类型（例如手机、座机或 VoIP）。有关更多信息，请参阅《Amazon Pinpoint 开发人员指南》中的在 Amazon Pinpoint 中验证电话号码。

设计时要考虑冗余

对于任务关键型消息收发程序，我们建议您在多个SNS AWS 区域中配置 Amazon。Amazon SNS 有几种可供选择 AWS 区域。有关 Amazon SNS 提供服务的地区的完整列表，请参阅AWS 一般参考。

您用于SMS发送消息的电话号码（包括短码、长码、免费电话DLC号码和 10 个号码）无法在各处复制。 AWS 区域因此，要SNS在多个地区使用亚马逊，您必须在要使用亚马逊的每个地区申请单独的电话号码SNS。例如，如果您使用短代码向美国的收件人发送短信，则需要在计划使用的每个短代码中申请单独 AWS 区域 的短代码。

在某些国家/地区，您还可以使用多种类型的电话号码来增加冗余。例如，在美国，您可以申请短码、10 个DLC号码和免费电话号码。这些电话号码中的每一种都采用不同的途径向收件人发消息。提供多种电话号码类型（相同 AWS 区域 或分布在多个电话号码中 AWS 区域）可提供额外的冗余层，这有助于提高弹性。

SMS限制和限制

有关限制和SMS限制，请参阅SMS《亚马逊 Pinpoint 用户指南》中亚马逊 Pinpoin t 中的限制和限制。

管理退出关键词

SMS收件人可以使用其设备通过回复关键字来选择退出消息。有关更多信息，请参阅 选择不接收消息 SMS。

CreatePool

使用`CreatePool`API操作创建新池并将指定的发起身份与池相关联。有关更多信息，请参阅 CreatePoolAmazon Pinpoint SMS 和语音。API

PutKeyword

使用该`PutKeyword`API操作在发起电话号码或电话池上创建或更新关键字配置。有关更多信息，请参阅 PutKeywordAmazon Pinpoint SMS 和语音。API

管理号码设置

您可以使用语音设置页面的 “号码设置” 部分中的SMS选项来管理您向 Su AWS pport 请求并分配给您帐户的专用短代码和长代码的设置。有关更多信息，请参阅《Amazon Pinpoint 用户指南》中的管理号码设置。

SMSAmazon 中的字符限制 SNS

一条SMS消息最多可以包含 140 字节的信息。单SMS封邮件中可以包含的字符数取决于消息中包含的字符类型。

如果您的消息仅使用 GSM03.38 字符集（也称为 GSM 7 位字母）中的字符，则最多可以包含 160 个字符。如果您的消息包含任何超出 GSM 03.38 字符集的字符，则最多可以包含 70 个字符。当您发送SMS消息时，Amazon SNS 会自动确定要使用的最有效的编码。

当消息包含的字符数超过最大字符数时，消息将拆分为多个部分。将消息拆分为多个部分时，每个部分都包含有关其前面的消息部分的其他信息。当接收人的设备接收以这种方式分隔的消息部分时，它使用此附加信息来确保所有消息部分都以正确的顺序显示。根据接收人的移动运营商和设备，多条消息可能会显示为单条消息或由单独消息组成的序列。因此，每个消息部分中的字符数减少到 153（对于仅包含 GSM 03.38 个字符的邮件）或 67（对于包含其他字符的邮件）。在发送消息之前，您可以使用SMS长度计算器工具估算消息包含多少部分，其中一些工具可在线获得。任何消息的最大支持大小为 1600 GSM 个字符或 630 个非GSM字符。有关吞吐量和邮件大小的更多信息，请参阅《亚马逊 Pinpoint 用户指南》中的 Amazon Pinp oin t 中的SMS字符限制。

要查看您发送的每条消息的消息部分的数量，您应首先启用事件流设置。当你这样做时，当消息传送到收件人的移动提供商时，Amazon 就会SNS生成一个_SMS.SUCCESS事件。_SMS.SUCCESS 事件记录包含名为 `attributes.number_of_message_parts` 的属性。此属性指定消息包含的消息部分的数量。

> **Important**
>
> 当您发送包含多个消息部分的消息时，您需要针对消息中包含的这些数量的消息部分付费。

GSM03.38 字符集

下表列出了 GSM 03.38 字符集中存在的所有字符。如果您所发送的消息只包含下表中显示的字符，那么该消息最多可以包含 160 个字符。

GSM03.38 个标准字符												
A	B	C	D	E	F	G	H	I	J	K	L	M
否	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z
à	Å	å	Ä	ä	Ç	É	é	è	ì	Ñ	ñ	ò
Ø	ø	Ö	ö	ù	Ü	ü	Æ	æ	ß	0	1	2
3	4	5	6	7	8	9	&	*	@	:	,	¤
$	=	!	>	#	-	¡	¿	(	<	%	.	+
£	?	"	)	§	;	'	/	_	¥	Δ	Φ	Γ
Λ	Ω	Π	Ψ	Σ	Θ	Ξ						

除了上表中显示的符号外，GSM03.38 字符集还包括多个符号。但是，这些字符中的每个字符都会算作两个字符，因为这些字符中还包含一个看不见的转义字符：

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

最后，GSM03.38 字符集还包括以下非打印字符：

- 空格字符。
- 换行控制，它表示一行文本的结束和另一行文本的开始。
- 回车控制，它会移动到一行文本的开头（通常跟在换行符后面）。
- 转义控制，它会自动添加到前一列表中的字符中。

示例消息

本节包含几个示例SMS消息。对于每个示例，此部分显示消息的字符总数以及消息部分的数量。

示例 1：仅包含 GSM 03.38 字母表字符的长消息

以下消息仅包含 GSM 03.38 字母表中的字符。

```
Hello Carlos. Your Example Corp. bill of $100 is now available. Autopay is scheduled for next Thursday, April 9. To view the details of your bill, go to https://example.com/bill1.
```

上述消息包含 180 个字符，因此必须将其拆分为多个消息部分。当一条消息分成多个消息部分时，每个部分可以包含 153 个 GSM 03.38 个字符。因此，此消息作为 2 个消息部分发送。

示例 2：包含多字节字符的消息

以下消息包含几个汉字，所有这些汉字都不是 GSM 03.38 字母。

```
##################################################·####1994#7#########
```

上述消息包含 71 个字符。但是，由于消息中的几乎所有字符都在 GSM 03.38 字母表之外，因此它是作为两个消息部分发送的。每个消息部分最多可包含 67 个字符。

示例 3：包含单个非GSM字符的消息

以下消息包含一个不属于 GSM 03.38 字母表的字符。在此示例中，该字符是一个右单引号 (’)，它是与常规撇号 (') 不同的字符。字处理应用程序（如 Microsoft Word）通常会自动用右单引号替换撇号。如果您在 Microsoft Word 中起草SMS邮件并将其粘贴到亚马逊中SNS，则应删除这些特殊字符并将其替换为撇号。

```
John: Your appointment with Dr. Salazar’s office is scheduled for next Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

上述消息包含 130 个字符。但是，由于它包含结尾的单引号字符（不是 GSM 03.38 字母的一部分），因此它作为两个消息部分发送。

如果将此消息中的结尾单引号字符替换为撇号（这是 GSM 03.38 字母表的一部分），则该消息将作为单个消息部分发送。

通过 Amazon 发送移动推送通知 SNS

借助 Amazon SNS，您可以直接向移动设备上的应用程序发送推送通知消息。发送到移动终端节点的推送通知消息可在移动应用程序中显示为消息提醒、徽章更新，甚至声音警报。

主题

用户通知的工作原理

使用以下受支持的推送通知服务之一将推送通知消息发送到移动设备和桌面：

- 亚马逊设备消息 (ADM)
- 适用于 iOS 和 Mac OS X 的 Apple 推送通知服务 (APNs)
- 百度云推送（百度）
- Firebase 云端消息传递 () FCM
- 适用于 Windows Phone 的微软推送通知服务 (MPNS)
- Windows 推送通知服务 (WNS)

推送通知服务（例如APNs和FCM）与注册使用其服务的每个应用程序和关联的移动设备保持连接。在应用程序和移动设备注册时，推送通知服务会返回设备令牌。Amazon SNS 使用设备令牌创建移动终

端节点，它可以向该终端节点发送直接推送通知消息。为了SNS让亚马逊与不同的推送通知服务进行通信，您需要向亚马逊提交您的推送通知服务凭证SNS以供您使用。有关更多信息，请参阅 用户通知流程概述。

除了发送直接推送通知消息外，您还可以使用 Amazon SNS 向订阅主题的移动终端节点发送消息。其概念与订阅其他终端节点类型（例如 Amazon SQS、HTTP /S、电子邮件和SMS主题）相同，如中所什么是亚马逊SNS？述。不同之处在于，Amazon 使用推送通知服务进行SNS通信，以便订阅的移动终端节点能够接收发送到该主题的推送通知消息。

用户通知流程概述

1. 为要支持的移动平台获取凭证和设备令牌。
2. 使用凭证通过 Amazon 创建平台应用程序对象 (`PlatformApplicationArn`) SNS。有关更多信息，请参阅 创建亚马逊SNS平台应用程序。
3. 使用返回的凭证从推送通知服务请求您的移动应用程序和设备的设备令牌。收到的令牌表示您的移动应用程序和设备。
4. 使用设备令牌和通过 Amazon 创建平台终端节点对象 (`EndpointArn`) SNS。`PlatformApplicationArn`有关更多信息，请参阅 设置用于移动通知的 Amazon SNS 平台终端节点。
5. 使用 `EndpointArn` 向移动设备上的应用发布消息。有关更多信息，请参阅直接向亚马逊SNS移动设备发送消息和《亚马逊简单通知服务API参考》API中的 “发布”。

在 Amazon 中设置移动应用程序 SNS

本节介绍如何使用 AWS Management Console 和中所述的信息Amazon SNS 用户通知的先决条件来设置移动应用程序。

主题

- Amazon SNS 用户通知的先决条件
- 创建亚马逊SNS平台应用程序
- 设置用于移动通知的 Amazon SNS 平台终端节点
- 将设备令牌与 Amazon 集成SNS以获取移动通知
- 亚马逊 SNS Apple 推送通知身份验证方法
- 亚马逊与 Firebase 云消息身份验证设置SNS集成
- 亚马逊SNS管理 Firebase 云消息终端节点

Amazon SNS 用户通知的先决条件

要开始使用 Amazon SNS 移动推送通知，您需要满足以下条件：

- 用于连接支持的推送通知服务之一的一组凭据：ADM、APNs、Baidu FCM、MPNS、或WNS。
- 移动应用程序和设备的设备令牌或注册 ID。
- Amazon SNS 配置为向移动终端节点发送推送通知消息。
- 注册并配置移动应用程序来使用支持的推送通知服务之一。

使用推送通知服务注册您的应用程序需要几个步骤。Amazon SNS 需要您向推送通知服务提供的一些信息，才能将直接推送通知消息发送到移动终端节点。通常而言，您需要连接推送通知服务所需的凭证、从推送通知服务获得的设备令牌或注册 ID（表示移动设备和移动应用程序），以及已注册推送通知服务的移动应用程序。

凭证的准确格式因移动平台而异，但在所有情况下，这些凭证必须在与平台建立连接时提交。为每个移动应用程序发布一组凭证，并且必须将其用于将消息发送到该应用程序的所有实例。

具体名称根据使用的推送通知服务而不同。例如，当APNs用作推送通知服务时，您需要设备令牌。或者，在使用时FCM，等效的设备令牌称为注册 ID。设备令牌或注册 ID 是由移动设备的操作系统发送到应用程序的字符串。它唯一标识运行在特定移动设备上的移动应用程序的实例，可以视为此应用程序/设备对的唯一标识符。

Amazon 将证书（以及其他一些设置）SNS存储为平台应用程序资源。设备令牌（以及一些其他设置）以被称为平台终端节点的对象来表示。每个平台终端节点属于一个特定平台应用程序，可以使用存储在其对应平台应用程序中的凭证与每个平台终端节点进行通信。

下面几节包括每个受支持推送通知服务的先决条件。获得必备信息后，您可以使用 AWS Management Console 或 Amazon SNS 移动推送发送推送通知消息APIs。有关更多信息，请参阅 用户通知流程概述。

创建亚马逊SNS平台应用程序

SNS要让 Amazon 直接或通过订阅主题向移动终端节点发送通知消息，您必须先创建平台应用程序。向注册应用程序后 AWS，下一步是为应用程序和移动设备创建终端节点。SNS然后，Amazon 使用终端节点向应用程序和设备发送通知消息。

要创建平台应用程序

1. 登录 Amazon SNS 控制台。

2. 在导航窗格中，选择 Mobile（移动），然后选择 Push notifications（推送通知）。
3. 在 Platform applications（平台应用程序）部分，选择 Create platform application（创建平台应用程序）。

   有关可以在其中创建移动应用程序的 AWS 区域的列表，请参阅Amazon SNS 移动应用程序支持的区域。
4. 对于 Application name（应用程序名称），输入一个名称来表示您的应用程序。

   应用程序名称只能由大写和小写ASCII字母、数字、下划线、连字符和句点组成。名称长度还必须只有 1–256 个字符。
5. 对于 Push notification platform（推送通知平台），选择注册该应用程序的平台，然后输入相应的凭证。

   > **Note**
   >
   > 如果您使用的是 Apple 推送通知服务 (APNs) 平台之一，则可以在基于令牌或证书的身份验证之间进行选择，然后选择选择文件将.p8 或 .p12 文件（从 Keychain Access 导出）上传到亚马逊。SNS
6. 然后选择 Create platform application（创建平台应用程序）。

   这会将应用程序注册到 AmazonSNS，Amazon 会为所选平台创建一个平台应用程序对象，然后返回相应的对象 PlatformApplicationArn。

设置用于移动通知的 Amazon SNS 平台终端节点

注册应用程序和移动设备的推送通知服务时，推送通知服务会返回设备令牌。Amazon SNS 使用设备令牌创建移动终端节点，它可以向该终端节点发送直接推送通知消息。有关更多信息，请参阅Amazon SNS 用户通知的先决条件 和用户通知流程概述。

此部分介绍了创建平台终端节点的推荐方法。

主题

- 创建平台终端节点
- 伪代码
- AWS SDK示例
- 故障排除

创建平台终端节点

要通过亚马逊向应用程序推送通知SNS，必须先SNS通过调用创建平台终端节点操作向亚马逊注册该应用程序的设备令牌。此操作将平台应用程序的 Amazon 资源名称 (ARN) 和设备令牌作为参数，并返回已创建ARN的平台终端节点的。

该CreatePlatformEndpoint操作执行以下操作：

- 如果平台终端节点已存在，则不重新创建。将现有平台终端节点ARN的信息返回给调用者。
- 如果存在具有相同设备令牌但不同设置的平台终端节点，则不重新创建。向调用方引发异常。
- 如果平台终端节点不存在，则创建它。向调用者返回新创建ARN的平台端点的值。

您不应在每次应用程序启动时立即调用创建平台终端节点操作，因为此方法并不总是提供正常工作的终端节点。例如，在相同设备上卸载并重新安装了应用程序，并且其终端节点已存在但被禁用时，会出现这种情况。成功的注册过程应该完成以下任务：

1. 确保此应用程序/设备组合存在平台终端节点。
2. 确保平台终端节点中的设备令牌是最新的有效设备令牌。
3. 确保平台终端节点已启用并且已准备好使用。

伪代码

以下伪代码介绍了在各种各样的开始条件中创建正常工作的、当前启用的平台终端节点的推荐做法。不论应用程序是否首次注册、此应用程序的平台终端节点是否已存在、平台终端节点是否已启用、是否具有正确的设备令牌等等，此方法均适用。连续多次调用该方法是安全的，因为它不会创建重复的平台终端节点；如果现有平台终端节点已是最新的并且已启用，也不会更改它。

```
retrieve the latest device token from the mobile operating system
if (the platform endpoint ARN is not stored)
  # this is a first-time registration
  call create platform endpoint
  store the returned platform endpoint ARN
endif

call get endpoint attributes on the platform endpoint ARN

if (while getting the attributes a not-found exception is thrown)
  # the platform endpoint was deleted
  call create platform endpoint with the latest device token
```

```
  store the returned platform endpoint ARN
else
  if (the device token in the endpoint does not match the latest one) or
      (get endpoint attributes shows the endpoint as disabled)
    call set endpoint attributes to set the latest device token and then enable the
 platform endpoint
  endif
endif
```

在应用程序希望注册或重新注册自身时，可以随时使用此方法。它也可以在通知 Amazon SNS 设备令牌变更时使用。在这种情况下，只需使用最新的设备令牌值调用操作即可。有关此方法需要说明的几点是：

- 在两种情况下可能会调用创建平台终端节点操作。它可能在一开始就被调用，因为应用程序不知道自己的平台端点ARN，就像在首次注册时发生的那样。如果初始获取端点属性操作调用失败并出现 “未找到” 异常，也会调用它，如果应用程序知道其端点ARN但它已被删除，就会发生这种情况。
- 调用获取终端节点属性操作来验证平台终端节点的状态，即使刚刚创建了平台终端节点。平台终端节点已存在但被禁用时会出现这种情况。在这种情况下，创建平台终端节点操作成功，但不启用平台终端节点，因此您必须在返回成功之前仔细检查平台终端节点的状态。

AWS SDK示例

以下代码演示如何使用提供的 Amazon SNS 客户端实现之前的伪代码。 AWS SDKs

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置AWS SDKs和凭据文件。

CLI

AWS CLI

创建平台应用程序端点

以下 `create-platform-endpoint` 示例使用指定令牌为指定平台应用程序创建端点。

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

输出：

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

有关更多信息，请参阅 移动推送API操作。

故障排除

使用过期设备令牌重复调用创建平台终端节点操作

特别是对于FCM端点，您可能会认为最好存储应用程序发布的第一个设备令牌，然后在每次应用程序启动时使用该设备令牌调用创建平台端点。这似乎是正确的，因为它使应用程序不必管理设备令牌的状态，而且 Amazon SNS 会自动将设备令牌更新为其最新值。但是，此解决方案存在多个严重问题：

- Amazon SNS 依靠来自的反馈FCM将过期的设备令牌更新为新的设备令牌。FCM将有关旧设备令牌的信息保留一段时间，但不能无限期保留。一旦FCM忘记了旧设备令牌和新设备令牌之间的连接，Amazon SNS 将无法再将存储在平台终端节点中的设备令牌更新为正确值；它只会禁用平台终端节点。
- 平台应用程序将包含与同一个设备令牌对应的多个平台终端节点。
- Amazon SNS 对可以从同一个设备令牌开始创建的平台终端节点数量设定配额。最终，新终端节点的创建将会失败，引发无效参数异常并显示以下错误消息：“此终端节点已注册到其他令牌。”

有关管理FCM终端节点的更多信息，请参阅亚马逊SNS管理 Firebase 云消息终端节点。

重新启用与无效设备令牌关联的平台终端节点

当移动平台（例如APNs或FCM）通知亚马逊SNS发布请求中使用的设备令牌无效时，亚马逊将SNS禁用与该设备令牌关联的平台终端节点。然后，Amazon SNS 将拒绝对该设备令牌的后续发布。虽然您可能会认为最好的方法是简单地重新启用平台终端节点并保持发布，但大多数情况下这样做不会有成效：发布的消息不会被传输，平台终端节点随后很快会再次被禁用。

这是因为与平台终端节点关联的设备令牌已真正无效。由于它不再对应于任何已安装的应用程序，因此以它为目标的传输操作不会成功。下次向其发布时，移动平台将再次通知亚马逊SNS设备令牌无效，亚马逊SNS将再次禁用该平台终端节点。

要重新启用已禁用的平台终端节点，该终端节点需要关联到有效的设备令牌（使用设置终端节点属性操作调用），然后再启用。只有这样，以该平台终端节点为目标的传输操作才会成功。要在不更新设备令牌的情况下重新启用平台终端节点，这种方法只有当与终端节点关联的设备令牌曾经无效但重新变得有效时才有起作用。例如，在相同移动设备上卸载、然后重新安装了某个应用程序并收到了相同的设备令牌时，会出现这种情况。上述方法会执行此操作，确保只有在验证与某个平台终端节点关联的设备令牌是最新可用的设备令牌时，才重新启用该平台终端节点。

将设备令牌与 Amazon 集成SNS以获取移动通知

当您首次使用通知服务（例如 Apple 推送通知服务 (APNs) 和 Firebase Cloud Messaging (FCM)）注册应用程序和移动设备时，通知服务会返回设备令牌或注册IDs信息。当您将设备令牌或注册添加IDs到 Amazon 时SNS，它们`PlatformApplicationArn`API将与一起使用，为应用程序和设备创建终端节点。当亚马逊SNS创建终端节点时，会返回一个`EndpointArn`。`EndpointArn`这就是Amazon SNS知道向哪个应用程序和移动设备发送通知消息的方式。

您可以使用以下方法向 Amazon IDs 添加设备令牌并SNS进行注册：

- AWS 使用手动添加单个令牌 AWS Management Console
- 使用上传多个代币 `CreatePlatformEndpoint` API
- 从将来会安装您的应用的设备注册令牌

手动添加设备令牌或注册 ID

1. 登录 Amazon SNS 控制台。
2. 选择移动，然后选择推送通知。
3. 在平台应用程序部分，选择您的应用程序，然后选择编辑。如果您尚未创建平台应用程序，请立即创建一个。有关如何执行此操作的说明，请参阅 创建亚马逊SNS平台应用程序。
4. 选择添加端点。
5. 在 Endpoint Token (终端节点令牌) 框中，根据通知服务输入令牌 ID 或注册 ID。例如，使用ADM和FCM输入注册 ID。
6. （可选）在 User Data（用户数据）中，输入要与终端节点关联的任意信息。亚马逊SNS不使用这些数据。数据必须采用 UTF -8 格式且小于 2KB。
7. 最后，选择 Add Endpoints (添加终端节点)。

   现在已经创建了终端节点，您可以直接向移动设备发送消息，也可以向订阅了某一主题的移动设备发送消息。

要上传多个令牌，请使用 **`CreatePlatformEndpoint`** API

以下步骤说明如何使用提供的示例 Java 应用程序（`bulkupload`软件包）将多个令牌（设备令牌或注册IDs）上传 AWS 到 Amazon SNS。您可以使用本示例应用帮助您开始上传现有令牌。

> **Note**
>
> 以下步骤使用 Eclipse Java。IDE这些步骤假设您已安装 AWS SDK for Java 并且拥有自己的 AWS 安全证书 AWS 账户。有关更多信息，请参阅 AWS SDK for Java。有关凭证的更多信息，请参阅《AWS 一般参考》中的如何获取安全凭证？

1. 下载并解压缩 snsmobilepush.zip 文件。
2. 在 Eclipse 中创建一个新的 Java 项目。
3. 将 SNSSamples 文件夹导入到新建的 Java 项目的顶级目录中。在 Eclipse 中，右键选择 Java 项目的名称，然后选择 Import (导入)，展开 General (常规)，依次选择 File System (文件系统)、Next (下一步)，浏览到 SNSSamples 文件夹，选择 OK (确定)，然后选择 Finish (完成)。
4. 下载 Ope n CSV 库的副本并将其添加到bulkupload包的构建路径中。
5. 打开 bulkupload 包中包含的 BulkUpload.properties 文件。
6. 将以下内容添加到 BulkUpload.properties 中：

   - 要向其添加终端节点的 ApplicationArn。
   - 包含令牌CSV的文件位置的绝对路径。
   - 为记录 Amazon 正确SNS解析的令牌goodTokens.csv和失败的令牌而创建的CSV文件（例如和badTokens.csv）的名称。
   - （可选）在包含标记CSV的文件中指定分隔符和引号的字符。
   - （可选）用于同时创建终端节点的线程数量。默认值为 1 个线程。

   完成后的 BulkUpload.properties 与下文类似：

   ```
   applicationarn:arn:aws:sns:us-west-2:111122223333:app/FCM/fcmpushapp
   csvfilename:C:\\mytokendirectory\\mytokens.csv
   goodfilename:C:\\mylogfiles\\goodtokens.csv
   badfilename:C:\\mylogfiles\\badtokens.csv
   delimiterchar:'
   quotechar:"
   numofthreads:5
   ```

7. 运行 BatchCreatePlatformEndpointSample .java 应用程序将令牌上传到亚马逊SNS。

在此示例中，为成功上传到 Amazon 的令牌创建的终端节点SNS将被登录到goodTokens.csv，而格式错误的令牌将被记录到badTokens.csv。此外，您应该会看到写入 Eclipse 控制台的STDOUT日志，其中包含类似于以下内容的内容：

```
<1>[SUCCESS] The endpoint was created with Arn arn:aws:sns:us-
west-2:111122223333:app/FCM/fcmpushapp/165j2214-051z-3176-b586-138o3d420071
<2>[ERROR: MALFORMED CSV FILE] Null token found in /mytokendirectory/mytokens.csv
```

从将来会安装您的应用的设备注册令牌

您可以使用下面两个选项之一：

- 使用 Amazon Cognito 服务：您的移动应用程序需要凭证才能创建与您的亚马逊SNS平台应用程序关联的终端节点。我们建议您使用会在一段时间后过期的临时凭证。对于大多数情况，我们建议您使用 Amazon Cognito 创建临时安全凭证。有关更多信息，请参阅 Amazon Cognito 开发人员指南。如果您希望在应用程序向亚马逊注册时收到通知SNS，则可以注册以接收将提供新终端节点的亚马逊SNS事件ARN。您也可以使用获取在 ListEndpointByPlatformApplication API Amazon 注册的终端节点的完整列表SNS。
- Use a proxy server：如果您的应用程序基础设施已经过设置，可使您的移动应用程序在每次安装时进行调用和注册，则可以继续使用此设置。您的服务器将充当代理，并将设备令牌以及您想要存储的任何用户数据传递给 Amazon SNS 移动推送通知。为此，代理服务器将SNS使用您的 AWS 证书连接到 Amazon，并使用CreatePlatformEndpointAPI调用上传令牌信息。系统将返回新创建的终端节点 Amazon 资源名称 (ARN)，您的服务器可以存储该终端节点，以便随后对亚马逊进行发布调用SNS。

亚马逊 SNS Apple 推送通知身份验证方法

您可以授权亚马逊SNS向您的 iOS 或 macOS 应用程序发送推送通知，方法是提供可识别您是应用程序开发者的信息。要进行身份验证，请在创建平台应用程序时提供密钥或者证书，您可以通过 Apple 开发人员账户获得这两项。

令牌签名密钥

亚马逊SNS用于签署 Apple 推送通知服务 (APNs) 身份验证令牌的私有签名密钥。

如果您提供签名密钥，Amazon 将SNS使用令牌对您发送APNs的每条推送通知进行身份验证。使用您的签名密钥，您可以向APNs生产环境和沙盒环境发送推送通知。

您的签名密钥不会过期，您可以将相同的签名密钥用于多个应用程序。有关更多信息，请参阅 Apple 网站 “开发者账户帮助” 部分中的APNs使用身份验证令牌进行通信。

证书

当您发送推送通知APNs时，Amazon SNS 用来进行身份验证的TLS证书。您可以从 Apple 开发人员账户获取该证书。

证书将在一年后过期。发生这种情况时，您必须创建新证书并将其提供给 Amazon SNS。有关更多信息，请参阅 Apple 开发者网站APNs上的与建立基于证书的连接。

使用管理控制台 AWS 管理APNs设置

1. 登录 Amazon SNS 控制台。
2. 在 Mobile（移动）下，选择 Push notifications（推送通知）。
3. 选择要编辑其APNs设置的应用程序，然后选择 “编辑”。
4. 在 Edit（编辑）页面中，对于 Authentication type（身份验证类型），选择 Token（令牌）或 Certificate（证书）。
5. 为证书或令牌签名密钥加载相应的凭证。您可以从 Apple 开发人员账户获取该信息。
6. 根据您选择的身份验证类型，执行以下操作之一：
   - 如果选择 Token（令牌），提供您的 Apple 开发人员账户中的以下信息。Amazon SNS 需要这些信息来构造身份验证令牌。
     - Signing key（签名密钥）– Apple 开发人员账户中的身份验证令牌签名密钥，您可以将其作为 .p8 文件下载。Apple 只允许您下载一次签名密钥。
     - Signing key ID（签名密钥 ID）– 分配给您的签名密钥的 ID。Amazon SNS 需要这些信息来构造身份验证令牌。要在你的 Apple 开发者帐户中找到此值，请选择 “证书IDs和配置文件”，然后在 “密钥” 部分中选择您的密钥。
     - Team identifier（团队标识符）– 分配给您的 Apple 开发人员账户团队的 ID。您可以在 Membership（会员资格）页面上找到此值。
     - Bundle identifier（服务包标识符）– 分配给您的应用程序的 ID。要找到此值，请选择 “证书IDs和配置文件”，在 “标识符” 部分IDs中选择 “应用程序”，然后选择您的应用程序。
   - 如果您选择 Certificate（证书），请提供以下信息：

- SSL证书-您的TLS证书的.p12 文件。在从 Apple 开发人员账户下载并安装证书之后，您可以从 Keychain Access 导出此文件。
- 证书密码 – 如果您向证书分配了密码，请在此处指定。
- 加载证书-选择加载证书以上传您的证书。

7. 完成后，选择 Save changes (保存更改)。

亚马逊与 Firebase 云消息身份验证设置SNS集成

本主题介绍如何从 Google 获取必需的 FCM API (HTTPv1) 凭据，以便与 AWS API、 AWS CLI 和一起使用。 AWS Management Console

主题

- 先决条件
- 使用管理FCM设置 API
- 使用管理FCM设置 CLI
- 使用控制台管理FCM设置

> **Important**
>
> 2023 年 6 月 20 日 — 谷歌弃用了 Firebase 云消息传递 (FCM) 的旧版。HTTP APIAmazon SNS 现在支持使用 FCM HTTP v1 API 向所有设备类型配送。我们建议您在 2024 年 6 月 1 API 日当天或之前将现有的移动推送应用程序迁移到最新的 FCM HTTP v1，以避免中断。
> 2024 年 1 月 18 日 — 亚马逊SNS推出对 FCM HTTP v1 的支持，API支持向安卓设备发送移动推送通知。
> 2024 年 3 月 26 日 — 亚马逊SNS支持 FCM HTTP v1 API 适用于苹果设备和 Webpush 目的地。我们建议您在 2024 年 6 月 1 API 日当天或之前将现有的移动推送应用程序迁移到最新的 FCM HTTP v1，以避免应用程序中断。

您可以通过提供SNS可识别您是应用程序开发者的信息，授权Amazon向您的应用程序发送推送通知。要进行身份验证，请在创建平台应用程序时提供API密钥或令牌。您可以从 Firebase 应用程序控制台获取以下信息：

API钥匙

API密钥是调用 Firebase 的旧版时使用的凭据。API谷歌APIs将于2024年6月20日移除Leg FCM acy。如果您当前使用API密钥作为平台凭证，则可以通过选择令牌作为选项并上传您的 Firebase 应用程序的关联JSON文件来更新平台凭证。

令牌

调用 HTTP v1 API 时使用短暂的访问令牌。这是 Firebase 在发送推送通知时API所建议的。为了生成访问令牌，Firebase 以私有密钥文件（也称为 service.json 文件）的形式为开发人员提供了一组凭证。

先决条件

您必须先获取 s FCM ervice.json 凭证，然后才能开始在亚马逊中管理FCM设置。SNS要获取您的 service.json 凭据，请参阅谷歌 Firebase 文档中的从旧版迁移FCMAPIs到 HTTP v1。

使用管理FCM设置 API

您可以使用创建FCM推送通知 AWS API。一个 AWS 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅AWS 一般参考 指南中的 Amazon 简单通知服务终端节点和配额。

将FCM推送通知与 Amazon SNS 主题一起创建 (AWS API)

使用密钥凭证时，`PlatformCredential` 为 `API key`。使用令牌凭证时，`PlatformCredential`是一个JSON格式化的私钥文件：

- `CreatePlatformApplication`

检索现有 Amazon SNS 主题的FCM凭证类型 ()AWS API

检索凭证类型 `"AuthenticationMethod": "Token"` 或 `"AuthenticationMethod": "Key"`：

- GetPlatformApplicationAttributes

为现有 Amazon SNS 主题设置FCM属性 (AWS API)

设置FCM属性：

- SetPlatformApplicationAttributes

使用管理FCM设置 CLI

您可以使用 AWS Command Line Interface (CLI) 创建FCM推送通知。一个 AWS 账户中 Amazon SNS 资源的数量和大小是有限的。有关更多信息，请参阅 Amazon Simple Notification Service 端点和配额。

将FCM推送通知与 Amazon SNS 主题一起创建 (AWS CLI)

使用密钥凭证时，`PlatformCredential` 为 `API key`。使用令牌凭证时，`PlatformCredential`是一个JSON格式化的私钥文件。使用时 AWS CLI，文件必须为字符串格式，并且必须忽略特殊字符。要正确格式化文件，Amazon SNS 建议使用以下命令`SERVICE_JSON=`jq @json <<< cat service.json``：

- create-platform-application

检索现有 Amazon SNS 主题的FCM凭证类型 ()AWS CLI

检索凭证类型 `"AuthenticationMethod": "Token"` 或 `"AuthenticationMethod": "Key"`：

- get-platform-application-attributes

为现有 Amazon SNS 主题设置FCM属性 (AWS CLI)

设置FCM属性：

- set-platform-application-attributes

使用控制台管理FCM设置

使用以下步骤输入您的应用程序用于连接的凭据FCM。

1. 登录 Amazon SNS 控制台。
2. 在 Mobile（移动）下，选择 Push notifications（推送通知）。
3. 选择现有FCM应用程序，然后选择 “编辑”。如果您尚未创建平台应用程序，请参阅创建亚马逊SNS平台应用程序。
4. 在编辑页面上，对于 Firebase Cloud Messaging 凭证，选择令牌或密钥。您可以从 Firebase 应用程序控制台获取以下信息。

- 如果您选择令牌，请上传有效的私有密钥文件。此文件的内容用于在发送通知时生成有效期很短的访问令牌。
- 如果您选择密钥，请输入 Google API 密钥。

5. 完成后，选择 Save changes (保存更改)。

相关主题

- 在亚马逊中使用谷歌 Firebase Cloud Messaging v1 有效负载 SNS

亚马逊SNS管理 Firebase 云消息终端节点

主题

- 管理和维护设备令牌
- 检测无效令牌
- 移除陈旧的代币

管理和维护设备令牌

您可以按照以下步骤确保移动应用程序推送通知的可送达性：

1. 将所有设备令牌、相应的 Amazon SNS 终端节点ARNs和时间戳存储在您的应用程序服务器上。
2. 移除所有陈旧的令牌并删除相应的 Amazon SNS 终端节点ARNs。

应用程序首次启动后，您将收到该设备的设备令牌（也称为注册令牌）。此设备令牌由设备的操作系统铸造，并与您的FCM应用程序相关联。收到此设备令牌后，您可以将其注册SNS为平台终端节点。我们建议您将设备令牌、Amazon SNS 平台终端节点ARN和时间戳保存到您的应用程序服务器或其他永久存储中，从而存储它们。要将您的FCM应用设置为检索和存储设备令牌，请参阅 Google 的 Firebase 文档中的检索和存储注册令牌。

维护 up-to-date 代币很重要。在以下情况下，用户的设备令牌可能会发生变化：

1. 移动应用程序将在新设备上恢复。
2. 用户卸载或更新应用程序。
3. 用户清除应用程序数据。

当您的设备令牌发生变化时，我们建议您使用新令牌更新相应的 Amazon SNS 终端节点。这样 SNS，Amazon 就可以继续与注册的设备进行通信。您可以通过在移动应用程序中实现以下伪代码来做到这一点。它描述了创建和维护已启用的平台端点的推荐做法。这种方法可以在每次移动应用程序启动时执行，也可以在后台作为计划任务执行。

伪代码

使用以下FCM伪代码管理和维护设备令牌。

```
retrieve the latest token from the mobile OS
if (endpoint arn not stored)
    # first time registration
    call CreatePlatformEndpoint
    store returned endpoint arn
endif

call GetEndpointAttributes on the endpoint arn

if (getting attributes encountered NotFound exception)
    #endpoint was deleted
    call CreatePlatformEndpoint
    store returned endpoint arn
else
    if (token in endpoint does not match latest) or
        (GetEndpointAttributes shows endpoint as disabled)
        call SetEndpointAttributes to set the
                    latest token and enable the endpoint
    endif
endif
```

要详细了解令牌更新要求，请参阅 Google 的 Firebase 文档中的定期更新令牌。

检测无效令牌

当向带有无效设备令牌的 FCM v1 终端节点发送消息时，Amazon SNS 将收到以下例外情况之一：

- UNREGISTERED(HTTP404) — 当亚马逊SNS收到此异常时，您将收到一个传送失败事件`InvalidPlatformToken`，且与终端节点关联的 of Platform 令牌无效。`FailureType` `FailureMessage`除此例外情况外，当交付失败时，Amazon SNS 将禁用您的平台终端节点。
- `INVALID_ARGUMENT`(HTTP400) — 当 Amazon SNS 收到此异常时，表示设备令牌或消息负载无效。有关更多信息，请参阅ErrorCode谷歌的 Firebase 文档。

由于在这两种情况下INVALID_ARGUMENT都可以退货，因此 Amazon SNS 将返回 o FailureType fInvalidNotification，而 o FailureMessage f 通知正文无效。当您收到此错误时，请验证您的有效载荷是否正确。如果正确，请验证设备令牌是否正确 up-to-date。除此例外情况外，当交付失败时，Amazon SNS 不会禁用您的平台终端节点。

您会遇到InvalidPlatformToken传送失败事件的另一种情况是，注册的设备令牌不属于尝试发送该消息的应用程序。在这种情况下，谷歌将返回一个 SENDER_ID_ 错误MISMATCH。除此例外情况外，当交付失败时，Amazon SNS 将禁用您的平台终端节点。

当您为应用程序设置交付状态日志 CloudWatch时API，可以查看从 FCM v1 收到的所有观察到的错误代码。

要接收应用程序的交付事件，请参阅可用应用程序事件。

移除陈旧的代币

一旦向端点设备传送消息开始失败，令牌就会被视为过时。Amazon SNS 将这些陈旧的令牌设置为您的平台应用程序的禁用终端节点。当您向已禁用的终端节点发布内容时，Amazon SNS 将返回一个带有 of Endpoin EventDeliveryFailure t 的事件EndpointDisabled，并且 a of FailureMessage E ndpoint 被禁用。FailureType要接收应用程序的交付事件，请参阅可用应用程序事件。

当您收到来自Amazon的此错误时SNS，您需要删除或更新平台应用程序中的陈旧令牌。

使用 Amazon SNS 发送移动推送通知

本部分描述如何发送移动推送通知消息。

主题

- 向主题发布
- 直接向亚马逊SNS移动设备发送消息
- 发布带有特定平台有效负载的 Amazon SNS 通知

向主题发布

您还可以使用 Amazon SNS 向订阅主题的移动终端节点发送消息。其概念与订阅其他终端节点类型（例如 Amazon SQS、HTTP /S、电子邮件和SMS主题）相同，如中所什么是亚马逊SNS？述。不同之处在于，亚马逊通过苹果推送通知服务 (APNS) 和谷歌 Firebase Cloud M SNS essaging () 等通知服务进行通信。FCM通过通知服务通信，订阅的移动终端节点可以接收发送给相应主题的通知。

直接向亚马逊SNS移动设备发送消息

您可以将 Amazon SNS 推送通知消息直接发送到代表移动设备上应用程序的终端节点。

发送直送消息

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择 Push notifications（推送通知）。
3. 例如，在移动推送通知页面的平台应用程序部分，选择应用程序的名称，例如 ***MyApp***.
4. 在 ***MyApp*** 页面上，在终端节点部分中，选择终端节点，然后选择发布消息。
5. 在 Publish message to endpoint (向终端节点发布消息) 页面上，输入将显示在移动设备上的应用程序中的消息，然后选择发布消息。

   Amazon SNS 将通知消息发送到平台通知服务，而平台通知服务又将消息发送给应用程序。

发布带有特定平台有效负载的 Amazon SNS 通知

您可以使用 AWS Management Console 或 Amazon SNS APIs 向移动设备发送包含特定平台负载的自定义消息。有关使用 Amazon 的信息 SNSAPIs，请参阅移动推送API操作和中的SNSMobilePush.java文件snsmobilepush.zip。

主题

- 发送JSON格式化的消息
- 发送平台特定的消息
- 在多个平台上向应用程序发送消息
- 将消息APNs作为警报或后台通知发送到
- 在亚马逊中使用谷歌 Firebase Cloud Messaging v1 有效负载 SNS

发送JSON格式化的消息

发送特定于平台的负载时，必须将数据格式化为JSON键值对字符串，并对引号进行转义。

以下示例显示了FCM平台的自定义消息。

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
```

```
}
```

发送平台特定的消息

除了以键-值对形式发送自定义数据之外，您还可以发送平台特定的键-值对。

以下示例显示了在FCM参数中包含参数time_to_live以及自定义数据键值对之collapse_key后的情况。FCM data

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

有关 Amazon 支持的每种推送通知服务支持的键值对的列表SNS，请参阅以下内容：

> **Important**
>
> 亚马逊SNS现在支持 Firebase 云消息 (FCM) HTTP v1，API用于向安卓设备发送移动推送通知。
> 2024 年 3 月 26 日 — 亚马逊SNS支持 FCM HTTP v1 API 适用于苹果设备和 Webpush 目的地。我们建议您在 2024 年 6 月 1 API 日当天或之前将现有的移动推送应用程序迁移到最新的 FCM HTTP v1，以避免应用程序中断。

- APNs文档中的@@ 有效载荷密钥参考
- 文档中的 Firebase 云消息HTTP协议 FCM
- 在ADM文档中发送消息

在多个平台上向应用程序发送消息

要向安装在多个平台（例如FCM和）设备上的应用程序发送消息APNs，您必须先将移动终端节点订阅到 Amazon 中的主题，SNS然后将消息发布到该主题。

以下示例显示了在APNs、FCM和ADM上发送到已订阅移动终端节点的消息：

```
{
```

```
  "default": "This is the default message which must be present when publishing a
 message to a topic. The default message will only be used if a message is not present
 for
one of the notification platforms.",
  "APNS": "{\"aps\":{\"alert\": \"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"} }",
  "GCM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}",
  "ADM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}"
}
```

将消息APNs作为警报或后台通知发送到

Amazon SNS 可以向APNs广告发送消息alert或background通知（有关更多信息，请参阅APNs文档中的向您的应用程序推送后台更新）。

- alertAPNs通知通过显示警报消息、播放声音或在应用程序的图标上添加徽章来通知用户。
- backgroundAPNs通知会被唤醒或指示您的应用程序根据通知的内容采取行动，而不通知用户。

指定自定义 APNs 标头值

我们建议使用 Amazon SNS Publish API 操作为AWS.SNS.MOBILE.APNS.PUSH_TYPE保留消息属性指定自定义值 AWS SDKs，或 AWS CLI。以下CLI示例content-available将指定主题的设置apns-push-typebackground为1和为。

```
aws sns publish \
--endpoint-url https://sns.us-east-1.amazonaws.com \
--target-arn arn:aws:sns:us-east-1:123456789012:endpoint/APNS_PLATFORM/MYAPP/1234a567-
bc89-012d-3e45-6fg7h890123i \
--message '{"APNS_PLATFORM":"{\"aps\":{\"content-available\":1}}"}' \
--message-attributes '{ \
  "AWS.SNS.MOBILE.APNS.TOPIC":
{"DataType":"String","StringValue":"com.amazon.mobile.messaging.myapp"}, \
  "AWS.SNS.MOBILE.APNS.PUSH_TYPE":{"DataType":"String","StringValue":"background"} \
  "AWS.SNS.MOBILE.APNS.PRIORITY":{"DataType":"String","StringValue":"5"}}', \
--message-structure json
```

根据有效载荷APNs推断推送类型标头

如果您未设置apns-push-typeAPNs标头，Amazon SNS 会background根据您JSON格式的APNs负载配置aps字典中的content-available密钥将标头设置为alert或依据。

> **Note**
>
> 尽管background标题可以设置为其他值，但亚马逊SNS只能alert推断apns-push-type标题。

- apns-push-type 设置为 alert
  - 如果 aps 字典包含设置为 1 的 content-available 和一个或多个触发用户交互的键。
  - 如果 aps 字典包含设置为 0 的 content-available 或如果 content-available 密钥不存在。
  - 如果 content-available 键的值不是整数或布尔值。
- apns-push-type 设置为 background
  - 如果 aps 字典仅包含设置为 1 的 content-available 且不包含触发用户交互的其他键。

> **Important**
>
> 如果 Amazon SNS 将原始配置对象APNs作为仅限后台的通知发送，则您必须在字典1中包含 content-available set to。aps尽管您可以包含自定义键，但 aps 字典不得包含触发用户交互的任何键（例如，警报、徽章或声音）。

下面是一个示例原始配置对象。

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

在此示例中，Amazon SNS 将消息apns-push-typeAPNs标题设置为background。当 Amazon SNS 检测到apn字典中包含设置为1（且不包含任何其他可以触发用户交互的密钥）时，它会将标题设置为。content-available background

在亚马逊中使用谷歌 Firebase Cloud Messaging v1 有效负载 SNS

亚马逊SNS支持使用 FCM HTTP v1 API 向安卓、iOS 和 Webpush 目的地发送通知。本主题提供了使用或 Amazon 发布移动推送通知时的有效负载结构示例SNSAPI。CLI

发送FCM通知时，您可以在有效负载中包含以下消息类型：

- 数据消息-数据消息由您的客户端应用程序处理，并包含自定义键值对。在构造数据消息时，必须包括以JSON对象为值的`data`密钥，然后输入您的自定义键值对。
- 通知消息或显示消息-通知消息包含一组由处理的预定义密钥FCMSDK。这些密钥因您要交付的设备类型而异。有关特定于平台的通知密钥的更多信息，请参阅以下内容：
  - 安卓通知密钥
  - APNS通知密钥
  - Webpush 通知密钥

有关FCM消息类型的更多信息，请参阅 Google 的 Firebase 文档中的消息类型。

目录

使用 FCM v1 有效负载结构发送消息

如果您是首次创建FCM应用程序，或者希望利用 v1 的功能，则可以选择发送 FCM v1 格式的有效负FCM载。为此，必须包含顶级密钥`fcmV1Message`。有关构建 FCM v1 有效负载的更多信息，请参阅 Google 的 Firebase 文档中的从旧版迁移FCMAPIs到 HTTP v1 和跨平台自定义消息。

FCM发送给 Amazon SNS 的 v1 示例有效负载：

> **Note**
>
> 使用 Amazon SNS 发布通知时，以下示例中使用的`GCM`密钥值必须编码为字符串。

```
{
  "GCM": "{
    \"fcmV1Message\": {
```

```
      \"validate_only\": false,
      \"message\": {
        \"notification\": {
          \"title\": \"string\",
          \"body\": \"string\"
        },
        \"data\": {
          \"dataGen\": \"priority message\"
        },
        \"android\": {
          \"priority\": \"high\",
          \"notification\": {
            \"body_loc_args\": [\"string\"],
            \"title_loc_args\": [\"string\"],
            \"sound\": \"string\",
            \"title_loc_key\": \"string\",
            \"title\": \"string\",
            \"body\": \"string\",
            \"click_action\": \"clicky_clacky\",
            \"body_loc_key\": \"string\"
          },
          \"data\": {
            \"dataAndroid\": \"priority message\"
          },
          \"ttl\": \"10023.32s\"
        },
        \"apns\": {
          \"payload\": {
            \"aps\": {
              \"alert\": {
                \"subtitle\": \"string\",
                \"title-loc-args\": [\"string\"],
                \"title-loc-key\": \"string\",
                \"loc-args\": [\"string\"],
                \"loc-key\": \"string\",
                \"title\": \"string\",
                \"body\": \"string\"
              },
              \"category\": \"Click\",
              \"content-available\": 0,
              \"sound\": \"string\",
              \"badge\": 5
            }
          }
```

```
        },
        \"webpush\": {
          \"notification\": {
            \"badge\": \"5\",
            \"title\": \"string\",
            \"body\": \"string\"
          },
          \"data\": {
            \"dataWeb\": \"priority message\"
          }
        }
      }
    }
  }"
}
```

发送JSON负载时，请务必在请求中包含该message-structure属性，并将其设置为json。

CLI示例：

```
aws sns publish --topic $TOPIC_ARN --message '{"GCM": "{\"fcmV1Message\": {\"message\":
{\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"android\":{\"priority
\":\"high\",\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"data\":
{\"customAndroidDataKey\":\"custom key value\"},\"ttl\":\"0s\"},\"apns\":{\"payload
\":{\"aps\":{\"alert\":{\"title\":\"string\", \"body\":\"string\"},\"content-
available\":1,\"badge\":5}}},\"webpush\":{\"notification\":{\"badge\":\"URL\",\"body
\":\"Test\"},\"data\":{\"customWebpushDataKey\":\"priority message\"}},\"data\":
{\"customGeneralDataKey\":\"priority message\"}}}", "default": "{\"notification\":
 {\"title\": \"test\"}"}' --region $REGION --message-structure json
```

有关发送 FCM v1 格式的负载的更多信息，请参阅 Google 的 Firebas e 文档中的以下内容：

- 从旧版迁移FCMAPIs到 HTTP v1
- 关于FCM消息
- REST资源：projects.messages

使用旧版有效载荷结构向 FCM v1 发送消息 API

迁移到 FCM v1 时，您无需更改用于旧版凭证的有效负载结构。Amazon SNS 会将您的有效负载转换为新的 FCM v1 有效载荷结构，然后发送给 Google。

输入消息有效载荷格式：

```
{
  "GCM": "{\"notification\": {\"title\": \"string\", \"body\": \"string\",
 \"android_channel_id\": \"string\", \"body_loc_args\": [\"string\"], \"body_loc_key\":
 \"string\", \"click_action\": \"string\", \"color\": \"string\", \"icon\": \"string
\", \"sound\": \"string\", \"tag\": \"string\", \"title_loc_args\": [\"string\"],
 \"title_loc_key\": \"string\"}, \"data\": {\"message\": \"priority message\"}}"
}
```

发送给谷歌的消息：

```
{
  "message": {
    "token": "***",
    "notification": {
      "title": "string",
      "body": "string"
    },
    "android": {
      "priority": "high",
      "notification": {
        "body_loc_args": [
          "string"
        ],
        "title_loc_args": [
          "string"
        ],
        "color": "string",
        "sound": "string",
        "icon": "string",
        "tag": "string",
        "title_loc_key": "string",
        "title": "string",
        "body": "string",
        "click_action": "string",
        "channel_id": "string",
        "body_loc_key": "string"
      },
      "data": {
        "message": "priority message"
      }
    },
    "apns": {
      "payload": {
```

```
        "aps": {
          "alert": {
            "title-loc-args": [
              "string"
            ],
            "title-loc-key": "string",
            "loc-args": [
              "string"
            ],
            "loc-key": "string",
            "title": "string",
            "body": "string"
          },
          "category": "string",
          "sound": "string"
        }
      }
    },
    "webpush": {
      "notification": {
        "icon": "string",
        "tag": "string",
        "body": "string",
        "title": "string"
      },
      "data": {
        "message": "priority message"
      }
    },
    "data": {
      "message": "priority message"
    }
  }
}
```

潜在风险

- 旧版到 v1 的映射不支持 Apple 推送通知服务 (APNS) `headers` 或`fcm_options`密钥。如果您想使用这些字段，请发送 FCM v1 有效负载。
- 在某些情况下，FCMv1 要求消息标头才能向您的APNs设备发送静默通知。如果您当前正在向APNs设备发送静默通知，则它们不适用于传统方法。相反，我们建议使用 FCM v1 有效负载以避免意外问题。要查找APNs标题列表及其用途，请参阅《Apple 开发者指南》APNs中的 “与之通信”。

- 如果您在发送通知时使用 TTL Amazon SNS 属性，则只会在该android字段中进行更新。如果要设置该TTLAPNS属性，请使用 FCM v1 有效负载。
- androidapns、和webpush键将被映射并填充所提供的所有相关密钥。例如，如果您提供title（这是所有三个平台共享的密钥），则 FCM v1 映射将使用您提供的标题填充所有三个平台。
- 一些平台间的共享密钥需要不同的值类型。例如，传递给的badge键需要apns一个整数值，而传递给的badge键webpush需要一个字符串值。在您提供badge密钥的情况下，FCMv1 映射将仅填充您为其提供有效值的密钥。

FCM配送失败事件

下表提供了与从 Google 收到的针对 FCM v1 通知请求的错误/状态代码相对应的 Amazon SNS 故障类型。当您为应用程序设置交付状态日志 CloudWatch 时API，可以查看从 FCM v1 收到的所有观察到的错误代码。

FCM错误/状态码	Amazon SNS 失败类型	失败消息	原因和缓解措施
UNREGISTERED	InvalidPlatformToken	与终端节点关联的平台令牌无效。	连接到您的终端节点的设备令牌已过时或无效。亚马逊SNS禁用了您的终端节点。将 Amazon SNS 终端节点更新为最新的设备令牌。
INVALID_ARGUMENT	InvalidNotification	通知正文无效。	设备令牌或消息负载可能无效。验证您的消息负载是否有效。如果消息负载有效，请将 Amazon SNS 终端节点更新为最新的设备令牌。
SENDER_ID_MISMATCH	InvalidPlatformToken	与终端节点关联的平台令牌无效。	与设备令牌关联的平台应用程序无权向设备令牌发送消息。验

FCM错误/状态码	Amazon SNS 失败类型	失败消息	原因和缓解措施
			证您在亚马逊SNS平台应用程序中使用的FCM凭证是否正确。
UNAVAILABLE	DependencyUnavailable	依赖关系不可用。	FCM无法及时处理请求。Amazon 执行的所有重试SNS都失败了。您可以将这些消息存储在死信队列 (DLQ) 中，稍后再重新驱动它们。
INTERNAL	UnexpectedFailure	意外故障；请联系 Amazon。失败短语 [内部错误]。	FCM服务器在尝试处理您的请求时遇到错误。Amazon 执行的所有重试SNS都失败了。您可以将这些消息存储在死信队列 (DLQ) 中，稍后再重新驱动它们。
THIRD_PARTY_AUTH_ERROR	InvalidCredentials	平台应用程序凭证无效。	无法发送针对 iOS 设备或 Webpush 设备的消息。验证您的开发和生产凭证是否有效。
QUOTA_EXCEEDED	Throttled	请求受 [gcm] 限制。	已超过消息速率配额、设备消息速率配额或主题消息速率配额。有关如何解决此问题的信息，请参阅 ErrorCodeGoogle 的 Firebase 文档中的。

FCM错误/状态码	Amazon SNS 失败类型	失败消息	原因和缓解措施
PERMISSION_DENIED	InvalidNotification	通知正文无效。	如果出现PERMISSION_DENIED 异常，则调用方（您的FCM应用程序）无权在负载中执行指定操作。导航到您的FCM控制台，并验证您的凭证是否启用了所需的API操作。

Amazon SNS 移动应用程序属性

亚马逊简单通知服务 (AmazonSNS) 支持记录推送通知消息的发送状态。配置应用程序属性后，从 Amazon 发送到移动终端节点的消息的 CloudWatch 日志条目将发送SNS到日志。记录消息传输状态有助于提供更好的业务洞察力，例如以下方面：

- 了解推送通知消息是否已从 Amazon 传送SNS到推送通知服务。
- 识别从推送通知服务发送给 Amazon 的响应SNS。
- 确定消息停留时间（发布时间戳与将消息转交给推送通知服务之间的时间差）。

要为消息传送状态配置应用程序属性，可以使用 AWS Management Console、 AWS 软件开发套件 (SDKs) 或查询API。

主题

- 使用配置留言传送状态属性 AWS Management Console
- Amazon SNS 消息传送状态 CloudWatch 日志示例
- 使用配置留言传送状态属性 AWS SDKs
- 平台响应代码

使用配置留言传送状态属性 AWS Management Console

1. 登录 Amazon SNS 控制台。

2. 在导航面板中，指向移动，并选择推送通知。
3. 从平台应用程序部分，选择包含要接收 CloudWatch 日志的终端节点的应用程序。
4. 选择 Application Actions (应用程序操作)，然后选择 Delivery Status (传输状态)。
5. 在 “交付状态” 对话框中，选择 “创建IAM角色”。

   然后，您将被重定向到IAM控制台。
6. 选择 “允许”，授予 Amazon 代表您使用 CloudWatch 日志的SNS写入权限。
7. 现在，返回 “传送状态” 对话框中，在 “成功样本百分比 (0-100)” 字段中输入一个数字，表示要接收 CloudWatch 日志的成功发送邮件的百分比。

   > **Note**
   >
   > 为消息传送状态配置应用程序属性后，所有失败的消息传送都会生成 CloudWatch 日志。

8. 最后，选择保存配置。现在，您将能够查看和解析包含消息传送状态的 CloudWatch 日志。有关使用的更多信息 CloudWatch，请参阅CloudWatch文档。

Amazon SNS 消息传送状态 CloudWatch 日志示例

为应用程序终端节点配置消息传送状态属性后，将生成 CloudWatch 日志。JSON格式的示例日志如下所示：

SUCCESS

```
{
  "status": "SUCCESS",
  "notification": {
    "timestamp": "2015-01-26 23:07:39.54",
    "messageId": "9655abe4-6ed6-5734-89f7-e6a6a42de02a"
  },
  "delivery": {
    "statusCode": 200,
    "dwellTimeMs": 65,
    "token": "Examplei7fFachkJ1xjlqT64RaBkcGHochmf1VQAr9k-
IBJtKjp7fedYPzEwT_Pq3Tu0lroqro1cwWJUvgkcPPYcaXCpPWmG3Bqn-
wiqIEzp5zZ7y_jsM0PKPxKhddCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw",
    "attempts": 1,
```

```
    "providerResponse": "{\"multicast_id\":5138139752481671853,\"success
\":1,\"failure\":0,\"canonical_ids\":0,\"results\":[{\"message_id\":
\"0:1422313659698010%d6ba8edff9fd7ecd\"}]}",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/FCM/FCMPushApp/
c23e42de-3699-3639-84dd-65f84474629d"
  }
}
```

FAILURE

```
{
  "status": "FAILURE",
  "notification": {
    "timestamp": "2015-01-26 23:29:35.678",
    "messageId": "c3ad79b0-8996-550a-8bfa-24f05989898f"
  },
  "delivery": {
    "statusCode": 8,
    "dwellTimeMs": 1451,
    "token": "examp1e29z6j5c4df46f80189c4c83fjcgf7f6257e98542d2jt3395kj73",
    "attempts": 1,
    "providerResponse": "NotificationErrorResponse(command=8, status=InvalidToken,
 id=1, cause=null)",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/APNS_SANDBOX/
APNSPushApp/986cb8a1-4f6b-34b1-9a1b-d9e9cb553944"
  }
}
```

有关推送通知服务响应代码的列表，请参阅平台响应代码。

使用配置留言传送状态属性 AWS SDKs

AWS SDKs提供了多种语言版本，用于APIs在 Amazon 中使用消息传送状态属性SNS。

以下 Java 示例说明如何使用为推送通知消息的消息传送状态配置应用程序属性。SetPlatformApplicationAttributes API您可以对消息传输状态使用以下属性：SuccessFeedbackRoleArn、FailureFeedbackRoleArn 和 SuccessFeedbackSampleRate。SuccessFeedbackRoleArn和FailureFeedbackRoleArn属性用于向 Amazon 授予代表您使用 CloudWatch 日志的SNS写入权限。SuccessFeedbackSampleRate 属性用于指定成功传输消息的采样率百分比 (0-100)。配置该FailureFeedbackRoleArn属性后，所有失败的消息传送都会生成 CloudWatch 日志。

```
SetPlatformApplicationAttributesRequest setPlatformApplicationAttributesRequest = new
 SetPlatformApplicationAttributesRequest();
Map<String, String> attributes = new HashMap<>();
attributes.put("SuccessFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("FailureFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("SuccessFeedbackSampleRate", "5");
setPlatformApplicationAttributesRequest.withAttributes(attributes);
setPlatformApplicationAttributesRequest.setPlatformApplicationArn("arn:aws:sns:us-
west-2:111122223333:app/FCM/FCMPushApp");
sns.setPlatformApplicationAttributes(setPlatformApplicationAttributesRequest);
```

有关 Java SDK 版的更多信息，请参阅入门 AWS SDK for Java。

平台响应代码

下面是推送通知服务响应代码链接列表：

推送通知服务	响应代码
亚马逊设备消息 (ADM)	请参阅ADM文档中的响应格式。
苹果推送通知服务 (APNs)	请参阅《本地和远程通知编程指南》APNs中的“与之通信” 中的 HTTP/2 响应。APNs
Firebase 云端消息传递 () FCM	请参阅 Firebase Cloud Messaging 文档中的下游消息错误响应代码
适用于 Windows Phone 的微软推送通知服务 (MPNS)	请参阅 Windows 8 开发文档中的 Windows Phone 8 的推送通知服务响应代码。
Windows 推送通知服务 (WNS)	请参阅 Windows 8 开发文档中推送通知服务请求和响应标头（Windows 运行时应用程序）中的“响应代码”。

针对移动SNS应用程序的 Amazon 应用程序事件通知

Amaz SNS on 支持在某些应用程序事件发生时触发通知。然后，您可以对该事件采取一些编程操作。您的应用程序必须支持推送通知服务，例如 Apple 推送通知服务 (APNs)、Firebase 云消息 (FCM) 和

Windows 推送通知服务 (WNS)。您可以使用 Amazon SNS 控制台设置应用程序事件通知 AWS CLI，或 AWS SDKs。

主题

可用应用程序事件

应用程序事件通知跟踪各个平台终端节点何时创建、删除、更新以及出现传输故障。以下是应用程序事件的属性名称。

属性名称	通知触发器
EventEndpointCreated	向应用程序添加新的平台终端节点。
EventEndpointDeleted	删除与应用程序关联的任何平台终端节点。
EventEndpointUpdated	与应用程序关联的平台终端节点的任何属性发生更改。
EventDeliveryFailure	向与应用程序关联的任何平台终端节点的传输操作发生永久性故障。 **Note** 要跟踪平台应用程序端的传输故障，需要为应用程序订阅消息传输状态事件。有关更多信息，请参阅使用 Amazon SNS 应用程序属性获取消息传送状态。

您可以将任何属性与应用程序关联，然后应用程序就可以接收这些事件通知。

发送移动推送通知

要发送应用程序事件通知，您需要为每种事件类型指定用于接收通知的主题。当 Amazon SNS 发送通知时，该主题可以将通知路由到将采取编程操作的终端节点。

> **Important**
>
> 高容量应用程序将创建大量的应用程序事件通知（例如，数万条），这会“淹没”供人们使用的终端节点，例如电子邮件、电话号码和移动应用程序。在向主题发送应用程序事件通知时，需要考虑以下指导原则：
>
> - 每个接收通知的主题都应仅包含对编程终端节点（例如HTTP终端节点、Amazon SQS 队列或 AWS Lambda 函数）的订阅。HTTPS
> - 要减少通知触发的处理量，请将每个主题的订阅数限制在很小的数目（例如，五个或更少）。

您可以使用 Amazon SNS 控制台、 AWS Command Line Interface (AWS CLI) 或发送应用程序事件通知 AWS SDKs。

AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板中，选择移动，推送通知。
3. 在移动推送通知页面的平台应用程序部分，选择一个应用程序，然后选择编辑。
4. 展开 Event notifications (事件通知) 部分。
5. 依次选择 Actions 和 Configure events。
6. 输入ARNs要用于以下事件的主题：
   - 已创建终端节点
   - 已删除终端节点
   - 已更新终端节点
   - 传输失败
7. 选择 Save changes（保存更改）。

AWS CLI

运行 set-platform-application-attributes 命令。

以下示例为所有四个应用程序事件设置了相同的 Amazon SNS 主题：

```
aws sns set-platform-application-attributes
```

```
--platform-application-arn arn:aws:sns:us-east-1:12345EXAMPLE:app/FCM/
MyFCMPlatformApplication
--attributes EventEndpointCreated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointDeleted="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointUpdated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventDeliveryFailure="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents"
```

AWS SDKs

SNSAPI使用 Amazon 向亚马逊提交SetPlatformApplicationAttributes请求，设置应用程序事件通知 AWS SDK。

有关 AWS SDK开发者指南和代码示例的完整列表，包括入门帮助和有关先前版本的信息，请参阅将 Amazon SNS 与 AWS SDK。

移动推送API操作

要使用亚马逊SNS移动推送APIs，您必须首先满足推送通知服务的先决条件，例如 Apple 推送通知服务 (APNs) 和 Firebase Cloud Messaging (FCM)。有关这些先决条件的更多信息，请参阅Amazon SNS 用户通知的先决条件。

要使用向移动应用程序和设备发送推送通知消息APIs，必须先使用返回PlatformApplicationArn属性的CreatePlatformApplication操作。然后 PlatformApplicationArn 使用 CreatePlatformEndpoint 属性，返回 EndpointArn 属性。之后，可以在 EndpointArn 操作中使用 Publish 属性将通知消息发送到移动应用程序和设备，也可以在 EndpointArn 操作中使用 Subscribe 属性订阅主题。有关更多信息，请参阅 用户通知流程概述。

Amazon的SNS移动推送APIs如下：

`CreatePlatformApplication`

> 为其中一个支持的推送通知服务（例如APNs和）创建平台应用程序对象FCM，设备和移动应用程序可以注册到该服务。返回 PlatformApplicationArn 操作所使用的 CreatePlatformEndpoint 属性。

CreatePlatformEndpoint

为受支持推送通知服务上的设备和移动应用程序创建终端节点。CreatePlatformEndpoint 使用从 PlatformApplicationArn 操作返回的 CreatePlatformApplication 属性。EndpointArn 属性是使用 CreatePlatformEndpoint 时返回的，它用在 Publish 操作中将通知消息发送到移动应用程序和设备。

CreateTopic

创建可以发布消息的主题。

DeleteEndpoint

删除一个受支持推送通知服务上的设备和移动应用程序的终端节点。

DeletePlatformApplication

删除平台应用程序数据元。

DeleteTopic

删除主题及其所有订阅。

GetEndpointAttributes

检索设备和移动应用程序的终端节点属性。

GetPlatformApplicationAttributes

检索平台应用程序数据元的属性。

ListEndpointsByPlatformApplication

列出受支持推送通知服务中的设备和移动应用程序的终端节点和终端节点属性。

ListPlatformApplications

列出受支持推送通知服务的平台应用程序数据元。

Publish

向主题的所有订阅终端节点发送通知消息。

SetEndpointAttributes

设置设备和移动应用程序的终端节点属性。

SetPlatformApplicationAttributes

设置平台应用程序数据元的属性。

`Subscribe`

准备通过向终端节点发送确认消息来订阅终端节点。要实际创建订阅，端点所有者必须使用确认消息中的令牌调用 ConfirmSubscription 操作。

`Unsubscribe`

删除订阅。

常见的 Amazon SNS 移动推送API错误

下表列出了 Amazon SNS APIs 为移动推送返回的错误。有关 Amazon 移动版推送SNSAPIs的更多信息，请参阅移动推送API操作。

错误	描述	HTTPS状态码	API行动
应用程序名称为空字符串	必需的应用程序名称设置为空。	400	`CreatePlatformApplication`
平台名称为空字符串	必需的平台名称设置为空。	400	`CreatePlatformApplication`
平台名称无效	为平台名称提供的或 out-of-range 值无效。	400	`CreatePlatformApplication`
APNs— 校长不是有效的证书	为APNs主体（即证书）提供的SSL证书无效。有关更多信息，请参阅CreatePlatformApplication《Amazon 简单通知服务API参考》。	400	`CreatePlatformApplication`
APNs— Principal 是有效的证书，但不是.pem 格式	为APNs主体（即证书）提供了非.pem 格式的有效证书。SSL	400	`CreatePlatformApplication`

错误	描述	HTTPS状态码	API行动
APNs— 主体是已过期的证书	为APNs主体提供了过期的证书，即SSL证书。	400	CreatePlatformApplication
APNs— 委托人不是 Apple 颁发的证书	为APNs委托人提供了非 Apple 颁发的证书，即SSL证书。	400	CreatePlatformApplication
APNs— 未提供本金	没有提供APNs委托人，即SSL证书。	400	CreatePlatformApplication
APNs— 未提供凭证	未提供作为私钥的APNs凭证。有关更多信息，请参阅CreatePlatformApplication《Amazon 简单通知服务API参考》。	400	CreatePlatformApplication
APNs— 凭证不是有效的.pem 格式	作为私钥的APNs凭证不是有效的.pem 格式。	400	CreatePlatformApplication
FCM— serverAPIKey 未提供	没有提供作为API密钥的FCM凭证。有关更多信息，请参阅CreatePlatformApplication《Amazon 简单通知服务API参考》。	400	CreatePlatformApplication
FCM— serverAPIKey 为空	作为密API钥的FCM凭据为空。	400	CreatePlatformApplication

错误	描述	HTTPS状态码	API行动
FCM— serverAPIKey 是一个空字符串	作为密API钥的FCM凭据为空。	400	CreatePlatformApplication
FCM— serverAPIKey 无效	作为密API钥的FCM凭证无效。	400	CreatePlatformApplication
ADM— 未提供客户机密信息	未提供必需的客户端密钥。	400	CreatePlatformApplication
ADM— clientsecret 是一个空字符串	客户端密钥所需的字符串为空。	400	CreatePlatformApplication
ADM— client_secret 为空字符串	客户端密钥所需的字符串为空。	400	CreatePlatformApplication
ADM— client_secret 无效	客户端密钥所需的字符串无效。	400	CreatePlatformApplication
ADM— client_id 为空字符串	客户端 ID 所需的字符串为空。	400	CreatePlatformApplication
ADM— clientId 未提供	未提供客户端 ID 所需的字符串。	400	CreatePlatformApplication
ADM— clientid 是一个空字符串	客户端 ID 所需的字符串为空。	400	CreatePlatformApplication

错误	描述	HTTPS状态码	API行动
ADM— 客户端 ID 无效	客户端 ID 所需的字符串无效。	400	CreatePlatformApplication
EventEndpointCreated ARN格式无效	EventEndpointCreated ARN格式无效。	400	CreatePlatformApplication
EventEndpointDeleted ARN格式无效	EventEndpointDeleted ARN格式无效。	400	CreatePlatformApplication
EventEndpointUpdated ARN格式无效	EventEndpointUpdated ARN格式无效。	400	CreatePlatformApplication
EventDeliveryAttemptFailure ARN格式无效	EventDeliveryAttemptFailure ARN格式无效。	400	CreatePlatformApplication
EventDeliveryFailure ARN格式无效	EventDeliveryFailure ARN格式无效。	400	CreatePlatformApplication
EventEndpointCreated 不是一个现有的话题	EventEndpointCreated 不是一个现有的主题。	400	CreatePlatformApplication
EventEndpointDeleted 不是一个现有的话题	EventEndpointDeleted 不是一个现有的主题。	400	CreatePlatformApplication
EventEndpointUpdated 不是一个现有的话题	EventEndpointUpdated 不是一个现有的主题。	400	CreatePlatformApplication

错误	描述	HTTPS状态码	API行动
EventDeliveryAttemptFailure 不是一个现有的话题	EventDeliveryAttemptFailure 不是一个现有的主题。	400	CreatePlatformApplication
EventDeliveryFailure 不是一个现有的话题	EventDeliveryFailure 不是一个现有的主题。	400	CreatePlatformApplication
平台ARN无效	平台ARN无效。	400	SetPlatformAttributes
ARN平台有效但不属于用户	平台ARN有效，但不属于用户。	400	SetPlatformAttributes
APNs— 校长不是有效的证书	为APNs主体（即证书）提供的SSL证书无效。有关更多信息，请参阅CreatePlatformApplication《Amazon 简单通知服务API参考》。	400	SetPlatformAttributes
APNs— Principal 是有效的证书，但不是.pem 格式	为APNs主体（即证书）提供了非.pem 格式的有效证书。SSL	400	SetPlatformAttributes
APNs— 主体是已过期的证书	为APNs主体提供了过期的证书，即SSL证书。	400	SetPlatformAttributes
APNs— 委托人不是 Apple 颁发的证书	为APNs委托人提供了非 Apple 颁发的证书，即SSL证书。	400	SetPlatformAttributes
APNs— 未提供本金	没有提供APNs委托人，即SSL证书。	400	SetPlatformAttributes

错误	描述	HTTPS状态码	API行动
APNs— 未提供凭证	未提供作为私钥的APNs凭证。有关更多信息，请参阅CreatePlatformApplication《Amazon 简单通知服务API参考》。	400	SetPlatformAttributes
APNs— 凭证不是有效的.pem 格式	作为私钥的APNs凭证不是有效的.pem 格式。	400	SetPlatformAttributes
FCM— serverAPIKey 未提供	没有提供作为API密钥的FCM凭证。有关更多信息，请参阅CreatePlatformApplication《Amazon 简单通知服务API参考》。	400	SetPlatformAttributes
FCM— serverAPIKey 是一个空字符串	作为密API钥的FCM凭据为空。	400	SetPlatformAttributes
ADM— clientId 未提供	未提供客户端 ID 所需的字符串。	400	SetPlatformAttributes
ADM— clientid 是一个空字符串	客户端 ID 所需的字符串为空。	400	SetPlatformAttributes
ADM— 未提供客户机密信息	未提供必需的客户端密钥。	400	SetPlatformAttributes
ADM— clientsecret 是一个空字符串	客户端密钥所需的字符串为空。	400	SetPlatformAttributes
EventEndpointUpdated ARN格式无效	EventEndpointUpdated ARN格式无效。	400	SetPlatformAttributes

错误	描述	HTTPS状态码	API行动
EventEndpointDeleted ARN格式无效	EventEndpointDeleted ARN格式无效。	400	SetPlatformAttributes
EventEndpointUpdated ARN格式无效	EventEndpointUpdated ARN格式无效。	400	SetPlatformAttributes
EventDeliveryAttemptFailure ARN格式无效	EventDeliveryAttemptFailure ARN格式无效。	400	SetPlatformAttributes
EventDeliveryFailure ARN格式无效	EventDeliveryFailure ARN格式无效。	400	SetPlatformAttributes
EventEndpointCreated 不是一个现有的话题	EventEndpointCreated 不是一个现有的主题。	400	SetPlatformAttributes
EventEndpointDeleted 不是一个现有的话题	EventEndpointDeleted 不是一个现有的主题。	400	SetPlatformAttributes
EventEndpointUpdated 不是一个现有的话题	EventEndpointUpdated 不是一个现有的主题。	400	SetPlatformAttributes
EventDeliveryAttemptFailure 不是一个现有的话题	EventDeliveryAttemptFailure 不是一个现有的主题。	400	SetPlatformAttributes
EventDeliveryFailure 不是一个现有的话题	EventDeliveryFailure 不是一个现有的主题。	400	SetPlatformAttributes
平台ARN无效	平台ARN无效。	400	GetPlatformApplicationAttributes

错误	描述	HTTPS状态码	API行动
ARN平台有效但不属于用户	ARN该平台有效，但不属于用户。	403	GetPlatformApplicationAttributes
指定的令牌无效	指定的令牌无效。	400	ListPlatformApplications
平台ARN无效	平台ARN无效。	400	ListEndpointsByPlatformApplication
ARN平台有效但不属于用户	ARN该平台有效，但不属于用户。	404	ListEndpointsByPlatformApplication
指定的令牌无效	指定的令牌无效。	400	ListEndpointsByPlatformApplication
平台ARN无效	平台ARN无效。	400	DeletePlatformApplication
ARN平台有效但不属于用户	ARN该平台有效，但不属于用户。	403	DeletePlatformApplication
平台ARN无效	平台ARN无效。	400	CreatePlatformEndpoint
ARN平台有效但不属于用户	ARN该平台有效，但不属于用户。	404	CreatePlatformEndpoint

错误	描述	HTTPS状态码	API行动
Token is not specified	未指定令牌。	400	CreatePlatformEndpoint
Token is not of correct length	令牌长度不正确。	400	CreatePlatformEndpoint
Customer User data is too large	在 UTF -8 编码中，客户用户数据的长度不能超过 2048 字节。	400	CreatePlatformEndpoint
终端节点ARN无效	终端节点ARN无效。	400	DeleteEndpoint
终ARN端节点有效但不属于用户	终端节点ARN有效，但不属于用户。	403	DeleteEndpoint
终端节点ARN无效	终端节点ARN无效。	400	SetEndpointAttributes
终ARN端节点有效但不属于用户	终端节点ARN有效，但不属于用户。	403	SetEndpointAttributes
Token is not specified	未指定令牌。	400	SetEndpointAttributes
Token is not of correct length	令牌长度不正确。	400	SetEndpointAttributes
Customer User data is too large	在 UTF -8 编码中，客户用户数据的长度不能超过 2048 字节。	400	SetEndpointAttributes
终端节点ARN无效	终端节点ARN无效。	400	GetEndpointAttributes
终ARN端节点有效但不属于用户	终端节点ARN有效，但不属于用户。	403	GetEndpointAttributes
目标ARN无效	目标ARN无效。	400	Publish

错误	描述	HTTPS状态码	API行动
目标有效ARN但不属于该用户	目标ARN有效，但不属于该用户。	403	Publish
消息格式无效	消息格式无效。	400	Publish
消息大小超过协议/端节点服务支持的范围	消息大小超过协议/端节点服务支持的范围。	400	Publish

对移动推送通知使用 Amazon 上线SNS时间消息属性

亚马逊简单通知服务 (AmazonSNS) 支持为移动推送通知消息设置生存时间 (TTL) 消息属性。除此之外，还TTL可以在亚马逊SNS消息正文中为支持此功能的移动推送通知服务（例如发送到 Android 时的亚马逊设备消息 (ADM) 和 Firebase 云消息 (FCM)）进行设置。

TTL消息属性用于指定有关消息的过期元数据。这允许您指定推送通知服务（例如 Apple 推送通知服务 (APNs) 或FCM）必须将消息传送到端点的时间长度。如果由于某种原因（例如移动设备已关闭）无法在指定范围内传送消息TTL，则该消息将被丢弃，并且不会再尝试传送该消息。要TTL在消息属性中指定，可以使用 AWS Management Console、 AWS 软件开发套件 (SDKs) 或查询API。

主题

- TTL推送通知服务的消息属性
- 用于确定的优先顺序 TTL
- TTL使用指定 AWS Management Console

TTL推送通知服务的消息属性

以下是推送通知服务的TTL消息属性列表，您可以在使用 AWS SDKs或查询时使用这些属性进行设置API：

推送通知服务	TTL消息属性
亚马逊设备消息 (ADM)	AWS.SNS.MOBILE.ADM.TTL
苹果推送通知服务 (APNs)	AWS.SNS.MOBILE.APNS.TTL

推送通知服务	TTL消息属性
Apple 推送通知服务沙盒 (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
百度云推送（百度）	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase 云端消息（发送到安卓系统FCM时）	`AWS.SNS.MOBILE.FCM.TTL`
Windows 推送通知服务 (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

每种推送通知服务的处理TTL方式都不同。Amazon SNS 提供了所有TTL推送通知服务的抽象视图，便于指定TTL。当您使用 AWS Management Console 来指定TTL（以秒为单位）时，您只需输入一次TTL值，然后 Amazon SNS 将在发布消息时计算每种选定推送通知服务的值。TTL

TTL是相对于发布时间的。在将推送通知消息传递给特定的推送通知服务之前，Amazon 会SNS计算推送通知的停留时间（从发布时间戳到移交给推送通知服务之前的时间），并将剩余的时间传递TTL给特定的推送通知服务。如果TTL比停留时间短，Amazon 将SNS不会尝试发布。

如果您TTL为推送通知消息指定 a，则该TTL值必须为正整数，除非的值对推送通知服务`0`有特定的含义，例如使用APNs和FCM（发送到 Android 时）。如果该TTL值设置为，`0`且推送通知服务没有具体含义`0`，则 Amazon SNS 将删除该消息。有关使用`0`时设置为的TTL参数的更多信息APNs，请参阅 Binary Provider API 文档中的表 A-3 远程通知项目标识符。

用于确定的优先顺序 TTL

Amazon SNS 用来确定推送通知消息TTL的优先顺序基于以下顺序，其中最小的数字优先级最高：

1. 消息属性 TTL
2. 消息正文 TTL
3. 推送通知服务默认TTL（因服务而异）
4. 亚马逊SNS默认TTL（4 周）

如果您为同一条消息设置了不同的TTL值（一个在消息属性中，另一个在消息正文中），Amazon SNS 将修改消息正文TTL中的值，使其与消息属性中TTL指定的值相匹配。

TTL使用指定 AWS Management Console

1. 登录 Amazon SNS 控制台。

2. 在导航面板中，选择移动，推送通知。
3. 在 Mobile push notifications (移动推送通知) 页面上的平台应用程序部分中，选择应用程序。
4. 在 ***MyApplication*** 页面的终端节点部分中，选择应用程序终端节点，然后选择发布消息。
5. 在消息详细信息部分，输入TTL（推送通知服务将消息传送到终端节点所需的秒数）。
6. 选择发布消息。

Amazon SNS 移动应用程序支持的区域

目前，您可以在以下区域内创建移动应用程序：

- 美国东部（俄亥俄）
- 美国东部（弗吉尼亚州北部）
- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈州）
- 非洲（开普敦）
- Asia Pacific (Hong Kong)
- 亚太地区（雅加达）
- 亚太地区（孟买）
- 亚太地区（大阪）
- 亚太地区（首尔）
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太地区（东京）
- 加拿大（中部）
- 欧洲地区（法兰克福）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- 欧洲地区（米兰）
- 欧洲地区（巴黎）
- 欧洲地区（斯德哥尔摩）
- 中东（巴林）

- 中东 (UAE)
- 南美洲（圣保罗）
- AWS GovCloud （美国西部）

管理 Amazon SNS 移动推送通知的最佳实践

本部分介绍可帮助您提升客户参与度的最佳实践。

终端节点管理

如果由于用户在设备上进行操作（例如，在设备上重新安装应用程序）导致设备令牌发生变化，或者证书更新影响了在特定 iOS 版本上运行的设备，则可能导致传送过程出现问题。Apple 推荐的最佳做法是在APNs每次启动应用程序时进行注册。

由于设备令牌不会在用户每次打开应用程序时发生变化，因此`CreatePlatformEndpoint`API可以使用 empotent。但是，如果令牌本身无效，或者端点有效但已禁用（例如，生产环境和沙盒环境不匹配），这可能会为同一设备引入重复项。

可以使用设备令牌管理机制，例如伪代码中的一种此类机制。

有关管理和维护 FCM v1 设备令牌的信息，请参阅亚马逊SNS管理 Firebase 云消息终端节点。

传送状态日志记录

要监控推送通知的发送状态，我们建议您为亚马逊SNS平台应用程序启用传送状态记录。这有助于您排查传送失败问题，因为日志包含从推送平台服务返回的提供商响应代码。有关启用传送状态记录的详细信息，请参阅如何访问推送通知的 Amazon SNS 主题传送日志？ 。

事件通知

要以事件驱动的方式管理终端节点，您可以利用事件通知功能。这允许配置的 Amazon SNS 主题向订阅者分发事件，例如 Lambda 函数，用于终端节点创建、删除、更新和交付失败等平台应用程序事件。

Amazon SNS 电子邮件订阅设置和管理

本页介绍如何使用 AWS Management Console、 AWS SDK for Java或通过电子邮件地址订阅亚马逊SNS主题 AWS SDK for .NET。

注意

- 您无法自定义电子邮件消息的正文。电子邮件传输功能旨在提供内部系统提示，而不是营销消息。
- 仅标准主题支持直接订阅电子邮件端点。
- 电子邮件传送吞吐量根据亚马逊SNS配额进行限制。

Important

要防止邮件列表收件人取消订阅 Amazon SNS 主题电子邮件的所有收件人，请参阅设置需要身份验证才能取消订阅 Support 的电子邮件订阅。 AWS

使用电子邮件地址订阅 Amazon SNS 主题 AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择订阅。
3. 在 Subscriptions（订阅）页面上，选择 Create subscription（创建订阅）。
4. 在 Create subscription（创建订阅）页上的 Details（详细信息）部分中，执行以下操作：
   a. 对于主题 ARN，选择主题的 Amazon 资源名称 (ARN)。
   b. 对于协议，选择电子邮件。
   c. 对于 Endpoint（终端节点），输入电子邮件地址。
   d. （可选）要配置筛选策略，请展开 Subscription filter policy（订阅筛选策略）部分。有关更多信息，请参阅 亚马逊SNS订阅筛选政策。
   e. （可选）要启用基于有效负载的筛选，请将 `Filter Policy Scope` 配置为 `MessageBody`。有关更多信息，请参阅 Amazon SNS 订阅筛选政策范围。
   f. （可选）要为订阅配置死信队列，请展开 Redrive policy (dead-letter queue)（重新驱动策略（死信队列））部分。有关更多信息，请参阅 Amazon SNS 死信队列。
   g. 选择创建订阅。

   控制台将创建订阅并打开订阅的 Details（详细信息）页面。

您必须先确认订阅，然后才能开始接收消息。

要确认订阅

1. 查看您的电子邮件收件箱，然后在 Amazon 发送的电子邮件中选择 “确认订阅” SNS。
2. Amazon 会SNS打开您的网络浏览器，并显示带有您的订阅 ID 的订阅确认信息。

使用电子邮件地址订阅 Amazon SNS 主题 AWS SDK

要使用 AWS SDK，必须使用您的凭据对其进行配置。有关更多信息，请参阅《工具参考指南》和《工具参考指南》中的共享配置AWS SDKs和凭据文件。

以下代码示例演示如何使用 `Subscribe`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
        /// <summary>
        /// Creates a new subscription to a topic.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object, used
        /// to create an Amazon SNS subscription.</param>
        /// <param name="topicArn">The ARN of the topic to subscribe to.</param>
        /// <returns>A SubscribeResponse object which includes the subscription
        /// ARN for the new subscription.</returns>
        public static async Task<SubscribeResponse> TopicSubscribeAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            SubscribeRequest request = new SubscribeRequest()
            {
```

```
                TopicArn = topicArn,
                ReturnSubscriptionArn = true,
                Protocol = "email",
                Endpoint = "recipient@example.com",
            };

            var response = await client.SubscribeAsync(request);

            return response;
        }
```

使用可选筛选条件为队列订阅主题。

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- 有关API详细信息，请参阅AWS SDK for .NET API参考资料中的订阅。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
```

```
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

为移动应用程序订阅主题。

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
```

```
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

为主题订阅 Lambda 函数。

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
```

```
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

为SQS队列订阅主题。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);

            Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
 "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
```

```
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

使用过滤器订阅主题。

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }
```

```
                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
```

```
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);
```

```
    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}
```

- 有关API详细信息，请参阅AWS SDK for C++ API参考资料中的订阅。

CLI

AWS CLI

订阅主题

以下 subscribe 命令将电子邮件地址订阅到指定主题。

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

输出：

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- 有关API详细信息，请参阅《AWS CLI 命令参考》中的 “订阅”。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用可选筛选条件为队列订阅主题。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
```

```
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- 有关API详细信息，请参阅AWS SDK for Go API参考资料中的订阅。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
 email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
```

```
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

为HTTP终端节点订阅主题。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive
 notifications.
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn()
 + "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

为主题订阅 Lambda 函数。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
```

```
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考资料中的订阅。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
```

```
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

为移动应用程序订阅主题。

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
```

```
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
 *                            when an application registers for notifications.
 */
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

为主题订阅 Lambda 函数。

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
```

```
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

为SQS队列订阅主题。

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
```

```
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

使用过滤器订阅主题。

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅AWS SDK for JavaScript API参考资料中的订阅。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

为主题订阅 Lambda 函数。

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- 有关API详细信息，请参阅订阅AWS SDK以获取 Kotlin API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

为HTTP终端节点订阅主题。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅AWS SDK for PHP API参考资料中的订阅。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
 subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
 number
                         (in E.164 format) for SMS messages, or an email address
 for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
```

```
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
 topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
 topic.arn
            )
            raise
        else:
            return subscription
```

- 有关API详细信息，请参阅订阅以获取 Python (Boto3) API 参考。AWS SDK

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end
```

```
  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info("Subscription created successfully.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅AWS SDK for Ruby API参考资料中的订阅。

Rust

SDK对于 Rust

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- 有关API详细信息，请参阅订阅AWS SDK以获取 Rust API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
   TRY.
       oo_result = lo_sns->subscribe(                       "oo_result is
returned for testing purposes."
               iv_topicarn = iv_topic_arn
               iv_protocol = 'email'
               iv_endpoint = iv_email_address
               iv_returnsubscriptionarn = abap_true
           ).
       MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
   ENDTRY.
```

- 有关API详细信息，请参阅订阅AWSSDK以供SAPABAPAPI参考。

SNS使用 Amazon 的代码示例 AWS SDKs

以下代码示例展示了如何将 Amazon SNS 与 AWS 软件开发套件 (SDK) 一起使用。

基础知识是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

开始使用

你好 Amazon SNS

以下代码示例展示了如何开始使用 Amazon SNS。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
{
    static async Task Main(string[] args)
```

```
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your
 topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 ListTopics” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
```

```
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_sns.cpp 源文件的代码。

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
```

```
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
```

```
                                    paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 ListTopics” 中的。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
```

```
    fmt.Printf("\t%v\n", *topic.TopicArn)
   }
  }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Go API参考 ListTopics” 中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
```

```
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 ListTopics” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

初始化SNS客户端，并在您的账户中列出主题。

```
import { SNSClient, paginateListTopics } from "@aws-sdk/client-sns";

export const helloSns = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SNSClient({});

  // You can also use `ListTopicsCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListTopicsCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedTopics = paginateListTopics({ client }, {});
  const topics = [];

  for await (const page of paginatedTopics) {
    if (page.Topics?.length) {
      topics.push(...page.Topics);
```

```
    }
  }

  const suffix = topics.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your
 account.`,
  );
  console.log(topics.map((t) => `  * ${t.TopicArn}`).join("\n"));
};
```

- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 ListTopics” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- 有关API详细信息，请参阅ListTopics中的 Kotlin AWS SDK API 参考。

代码示例

- Amazon SNS 使用的基本示例 AWS SDKs
  - 你好 Amazon SNS
  - Amazon SNS 使用的操作 AWS SDKs
    - 与 AWS SDK或CheckIfPhoneNumberIsOptedOut一起使用 CLI
    - 与 AWS SDK或ConfirmSubscription一起使用 CLI
    - 与 AWS SDK或CreateTopic一起使用 CLI
    - 与 AWS SDK或DeleteTopic一起使用 CLI
    - 与 AWS SDK或GetSMSAttributes一起使用 CLI
    - 与 AWS SDK或GetTopicAttributes一起使用 CLI
    - 与 AWS SDK或ListPhoneNumbersOptedOut一起使用 CLI
    - 与 AWS SDK或ListSubscriptions一起使用 CLI
    - 与 AWS SDK或ListTopics一起使用 CLI
    - 与 AWS SDK或Publish一起使用 CLI
    - 与 AWS SDK或SetSMSAttributes一起使用 CLI
    - 与 AWS SDK或SetSubscriptionAttributes一起使用 CLI
    - 与 AWS SDK或SetSubscriptionAttributesRedrivePolicy一起使用 CLI
    - 与 AWS SDK或SetTopicAttributes一起使用 CLI
    - 与 AWS SDK或Subscribe一起使用 CLI
    - 与 AWS SDK或TagResource一起使用 CLI
    - 与 AWS SDK或Unsubscribe一起使用 CLI

Amazon SNS 使用的基本示例 AWS SDKs

以下代码示例展示了如何使用 Amazon 简单通知服务的基础知识 AWS SDKs。

示例

- 与 AWS SDK或ListSubscriptions一起使用 CLI
- 与 AWS SDK或ListTopics一起使用 CLI
- 与 AWS SDK或Publish一起使用 CLI
- 与 AWS SDK或SetSMSAttributes一起使用 CLI
- 与 AWS SDK或SetSubscriptionAttributes一起使用 CLI
- 与 AWS SDK或SetSubscriptionAttributesRedrivePolicy一起使用 CLI
- 与 AWS SDK或SetTopicAttributes一起使用 CLI
- 与 AWS SDK或Subscribe一起使用 CLI
- 与 AWS SDK或TagResource一起使用 CLI
- 与 AWS SDK或Unsubscribe一起使用 CLI

你好 Amazon SNS

以下代码示例展示了如何开始使用 Amazon SNS。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();
```

```
        Console.WriteLine($"Hello Amazon SNS! Following are some of your
 topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 ListTopics” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
```

```
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_sns.cpp 源文件的代码。

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
```

```
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
```

```
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 ListTopics” 中的。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
```

```
 }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Go API参考 ListTopics” 中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 ListTopics” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

初始化SNS客户端，并在您的账户中列出主题。

```
import { SNSClient, paginateListTopics } from "@aws-sdk/client-sns";

export const helloSns = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SNSClient({});

  // You can also use `ListTopicsCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListTopicsCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedTopics = paginateListTopics({ client }, {});
  const topics = [];

  for await (const page of paginatedTopics) {
    if (page.Topics?.length) {
      topics.push(...page.Topics);
    }
  }
```

```
  const suffix = topics.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your
 account.`,
  );
  console.log(topics.map((t) => `  * ${t.TopicArn}`).join("\n"));
};
```

- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 ListTopics” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
```

```
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- 有关API详细信息，请参阅ListTopics中的 Kotlin AWS SDK API 参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

Amazon SNS 使用的操作 AWS SDKs

以下代码示例演示了如何使用执行各个 Amazon SNS 操作 AWS SDKs。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

这些摘录称为 Amazon SNSAPI，是大型程序的代码摘录，这些程序必须在上下文中运行。您可以在Amazon SNS 使用的场景 AWS SDKs 中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅《Amazon 简单通知服务API参考》。

示例

- 与 AWS SDK或CheckIfPhoneNumberIsOptedOut一起使用 CLI
- 与 AWS SDK或ConfirmSubscription一起使用 CLI
- 与 AWS SDK或CreateTopic一起使用 CLI
- 与 AWS SDK或DeleteTopic一起使用 CLI
- 与 AWS SDK或GetSMSAttributes一起使用 CLI
- 与 AWS SDK或GetTopicAttributes一起使用 CLI
- 与 AWS SDK或ListPhoneNumbersOptedOut一起使用 CLI
- 与 AWS SDK或ListSubscriptions一起使用 CLI
- 与 AWS SDK或ListTopics一起使用 CLI
- 与 AWS SDK或Publish一起使用 CLI
- 与 AWS SDK或SetSMSAttributes一起使用 CLI
- 与 AWS SDK或SetSubscriptionAttributes一起使用 CLI
- 与 AWS SDK或SetSubscriptionAttributesRedrivePolicy一起使用 CLI

- 与 AWS SDK或SetTopicAttributes一起使用 CLI
- 与 AWS SDK或Subscribe一起使用 CLI
- 与 AWS SDK或TagResource一起使用 CLI
- 与 AWS SDK或Unsubscribe一起使用 CLI

与 AWS SDK或**CheckIfPhoneNumberIsOptedOut**一起使用 CLI

以下代码示例演示如何使用 CheckIfPhoneNumberIsOptedOut。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to use the Amazon Simple Notification Service
/// (Amazon SNS) to check whether a phone number has been opted out.
/// </summary>
public class IsPhoneNumOptedOut
{
    public static async Task Main()
    {
        string phoneNumber = "+15551112222";

        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        await CheckIfOptedOutAsync(client, phoneNumber);
    }
```

```
        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 CheckIfPhoneNumberIsOptedOut” 中的。

CLI

AWS CLI

要查看SMS留言，请选择退出电话号码

以下check-if-phone-number-is-opted-out示例检查指定的电话号码是否已选择不接收来自当前 AWS 账户的SMS消息。

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

输出：

```
{
    "isOptedOut": false
}
```

- 有关API详细信息，请参阅 “CheckIfPhoneNumberIsOptedOut AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
```

```
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 CheckIfPhoneNumberIsOptedOut” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";
```

```
import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 CheckIfPhoneNumberIsOptedOut” 中的。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 CheckIfPhoneNumberIsOptedOut” 中的。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**ConfirmSubscription**一起使用 CLI

以下代码示例演示如何使用 ConfirmSubscription。

CLI

AWS CLI

确认订阅

以下confirm-subscription命令完成订阅名my-topic为的SNS主题时启动的确认过程。--token 参数来自发送到订阅调用中指定的通知端点的确认消息。

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

输出：

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- 有关API详细信息，请参阅 “ConfirmSubscription AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionToken> <topicArn>

                Where:
                   subscriptionToken - A short-lived token sent to an endpoint
 during the Subscribe action.
                   topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
 String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
 ConfirmSubscriptionRequest.builder()
                    .token(subscriptionToken)
```

```
                    .topicArn(topicArn)
                    .build();

            ConfirmSubscriptionResponse result =
 snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                    + result.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 ConfirmSubscription” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                         that are not AWS services (HTTP/S, email) need to be
 confirmed.
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    // A subscription only needs to be confirmed if the endpoint type is
    // HTTP/S, email, or in another AWS account.
    new ConfirmSubscriptionCommand({
      Token: token,
      TopicArn: topicArn,
      // If this is true, the subscriber cannot unsubscribe while
 unauthenticated.
      AuthenticateOnUnsubscribe: "false",
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。

- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 ConfirmSubscription” 中的。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 ConfirmSubscription” 中的。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**CreateTopic**一起使用 CLI

以下代码示例演示如何使用 `CreateTopic`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建并发布到FIFO主题
- 将消息发布到队列

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用特定的名称创建主题。

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
```

```
    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

创建一个具有名称、特定属FIFO性和重复数据消除属性的新主题。

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }

        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 CreateTopic” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 CreateTopic” 中的。

CLI

AWS CLI

创建 SNS 主题

以下create-topic示例创建了一个名为SNS的主题my-topic。

```
aws sns create-topic \
    --name my-topic
```

输出：

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

有关更多信息，请参阅《 AWS 命令行界面用户指南》SNS中的在 Amazon SQS 和 Amazon 上使用AWS命令行界面。

- 有关API详细信息，请参阅 “CreateTopic AWS CLI命令参考”。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
```

```
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- 有关API详细信息，请参阅 “AWS SDK for Go API参考 CreateTopic” 中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
```

```
                    topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 CreateTopic” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
```

```
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 CreateTopic” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- 有关API详细信息，请参阅CreateTopic中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 CreateTopic” 中的。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- 有关API详细信息，请参阅CreateTopic中的 AWS SDKPython (Boto3) API 参考。

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# This class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
```

```
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for Ruby API参考 CreateTopic” 中的。

Rust

SDK对于 Rust

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- 有关API详细信息，请参见CreateTopic中的 Rust AWS SDK API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   TRY.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
is returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- 有关API详细信息，请参阅CreateTopic中的AWS SDK以供SAPABAPAPI参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**DeleteTopic**一起使用 CLI

以下代码示例演示如何使用 DeleteTopic。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 将消息发布到队列

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

按主题删除主题ARN。

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 DeleteTopic” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 DeleteTopic” 中的。

CLI

AWS CLI

删除SNS主题

以下delete-topic示例删除了指定的SNS主题。

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

此命令不生成任何输出。

- 有关API详细信息，请参阅 “DeleteTopic AWS CLI命令参考”。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- 有关API详细信息，请参阅 “AWS SDK for Go API参考 DeleteTopic” 中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 DeleteTopic” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 DeleteTopic” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- 有关API详细信息，请参阅DeleteTopic中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 DeleteTopic” 中的。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- 有关API详细信息，请参阅DeleteTopic中的 AWS SDKPython (Boto3) API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅DeleteTopic中的AWS SDK以供SAPABAPAPI参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**GetSMSAttributes**一起使用 CLI

以下代码示例演示如何使用 `GetSMSAttributes`。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Retrieve the default settings for sending SMS messages from your AWS account
 by using
//! Amazon Simple Notification Service (Amazon SNS).
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
            request);
```

```
    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for C++ API参考资料》中的 G。

CLI

AWS CLI

列出默认SMS消息属性

以下get-sms-attributes示例列出了发送SMS消息的默认属性。

```
aws sns get-sms-attributes
```

输出：

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
```

```
    }
}
```

- 有关API详细信息，请参阅etSMSAttributes《AWS CLI 命令参考》中的 G。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
```

```
                   topicArn - The ARN of the topic from which to retrieve
attributes.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
```

```
    }
}
```

- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for Java 2.x API参考资料》中的 G。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { GetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const getSmsAttributes = async () => {
  const response = await snsClient.send(
    // If you have not modified the account-level mobile settings of SNS,
    // the DefaultSMSType is undefined. For this example, it was set to
    // Transactional.
    new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
  );

  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   attributes: { DefaultSMSType: 'Transactional' }
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for JavaScript API参考资料》中的G。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */
```

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for PHP API参考资料》中的 G。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**GetTopicAttributes**一起使用 CLI

以下代码示例演示如何使用 `GetTopicAttributes`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    using System;
    using System.Collections.Generic;
```

```
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;

    /// <summary>
    /// This example shows how to retrieve the attributes of an Amazon Simple
    /// Notification Service (Amazon SNS) topic.
    /// </summary>
    public class GetTopicAttributes
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
west-2:000000000000:ExampleSNSTopic";
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var attributes = await GetTopicAttributesAsync(client, topicArn);
            DisplayTopicAttributes(attributes);
        }

        /// <summary>
        /// Given the ARN of the Amazon SNS topic, this method retrieves the
 topic
        /// attributes.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the attributes for the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic for which to retrieve
        /// the attributes.</param>
        /// <returns>A Dictionary of topic attributes.</returns>
        public static async Task<Dictionary<string, string>>
 GetTopicAttributesAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            var response = await client.GetTopicAttributesAsync(topicArn);

            return response.Attributes;
        }

        /// <summary>
        /// This method displays the attributes for an Amazon SNS topic.
        /// </summary>
        /// <param name="topicAttributes">A Dictionary containing the
```

```
        /// attributes for an Amazon SNS topic.</param>
        public static void DisplayTopicAttributes(Dictionary<string, string>
 topicAttributes)
        {
            foreach (KeyValuePair<string, string> entry in topicAttributes)
            {
                Console.WriteLine($"{entry.Key}: {entry.Value}\n");
            }
        }
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 GetTopicAttributes” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 GetTopicAttributes” 中的。

CLI

AWS CLI

检索主题的属性

以下 get-topic-attributes 示例将显示指定主题的属性。

```
aws sns get-topic-attributes \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

输出：

```
{
    "Attributes": {
        "SubscriptionsConfirmed": "1",
        "DisplayName": "my-topic",
        "SubscriptionsDeleted": "0",
        "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy
\":{\"minDelayTarget\":20,\"maxDelayTarget\":20,\"numRetries\":3,
\"numMaxDelayRetries\":0,\"numNoDelayRetries\":0,\"numMinDelayRetries\":0,
\"backoffFunction\":\"linear\"},\"disableSubscriptionOverrides\":false}}",
        "Owner": "123456789012",
```

```
        "Policy": "{\"Version\":\"2008-10-17\",\"Id\":\"__default_policy_ID
\",\"Statement\":[{\"Sid\":\"__default_statement_ID\",\"Effect\":
\"Allow\",\"Principal\":{\"AWS\":\"*\"},\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",\"SNS:DeleteTopic\",\"SNS:GetTopicAttributes
\",\"SNS:Publish\",\"SNS:RemovePermission\",\"SNS:AddPermission\",
\"SNS:SetTopicAttributes\"],\"Resource\":\"arn:aws:sns:us-west-2:123456789012:my-
topic\",\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":
\"0123456789012\"}}}]}",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
        "SubscriptionsPending": "0"
    }
}
```

- 有关API详细信息，请参阅 “GetTopicAttributes AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to look up.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Getting attributes for a topic with name: " +
 topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String
 topicArn) {
        try {
            GetTopicAttributesRequest request =
 GetTopicAttributesRequest.builder()
                    .topicArn(topicArn)
                    .build();

            GetTopicAttributesResponse result =
 snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
 result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                    + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 GetTopicAttributes” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { GetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
```

```
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDela
{"headerContentType":"text/plain; charset=UTF-8"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 GetTopicAttributes” 中的。

SDK对于 JavaScript (v2)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

导入SDK和客户端模块并调用API。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });
```

```
// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
  .promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 GetTopicAttributes” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- 有关API详细信息，请参阅GetTopicAttributes中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 GetTopicAttributes” 中的。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purposes. "
        DATA(lt_attributes) = oo_result->get_attributes( ).
        MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅GetTopicAttributes中的AWS SDK以供SAPABAPAPI参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**ListPhoneNumbersOptedOut**一起使用 CLI

以下代码示例演示如何使用 `ListPhoneNumbersOptedOut`。

CLI

AWS CLI

列出退出SMS留言

以下`list-phone-numbers-opted-out`示例列出了选择不接收SMS消息的电话号码。

```
aws sns list-phone-numbers-opted-out
```

输出：

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- 有关API详细信息，请参阅 “ListPhoneNumbersOptedOut AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
```

```
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 ListPhoneNumbersOptedOut” 中的。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
```

```
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 ListPhoneNumbersOptedOut” 中的。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**ListSubscriptions**一起使用 CLI

以下代码示例演示如何使用 `ListSubscriptions`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example will retrieve a list of the existing Amazon Simple
    /// Notification Service (Amazon SNS) subscriptions.
    /// </summary>
    public class ListSubscriptions
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            Console.WriteLine("Enter a topic ARN to list subscriptions for a
specific topic, " +
                              "or press Enter to list subscriptions for all
topics.");
            var topicArn = Console.ReadLine();
            Console.WriteLine();

            var subscriptions = await GetSubscriptionsListAsync(client,
topicArn);

            DisplaySubscriptionList(subscriptions);
        }

        /// <summary>
```

```
        /// Gets a list of the existing Amazon SNS subscriptions, optionally by
 specifying a topic ARN.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to obtain the list of subscriptions.</param>
        /// <param name="topicArn">The optional ARN of a specific topic. Defaults
 to null.</param>
        /// <returns>A list containing information about each subscription.</
returns>
        public static async Task<List<Subscription>>
 GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string
 topicArn = null)
        {
            var results = new List<Subscription>();

            if (!string.IsNullOrEmpty(topicArn))
            {
                var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                    new ListSubscriptionsByTopicRequest()
                    {
                        TopicArn = topicArn,
                    });

                // Get the entire list using the paginator.
                await foreach (var subscription in paginateByTopic.Subscriptions)
                {
                    results.Add(subscription);
                }
            }
            else
            {
                var paginateAllSubscriptions =
 client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

                // Get the entire list using the paginator.
                await foreach (var subscription in
 paginateAllSubscriptions.Subscriptions)
                {
                    results.Add(subscription);
                }
            }

            return results;
        }
```

```
        /// <summary>
        /// Display a list of Amazon SNS subscription information.
        /// </summary>
        /// <param name="subscriptionList">A list containing details for existing
        /// Amazon SNS subscriptions.</param>
        public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
        {
            foreach (var subscription in subscriptionList)
            {
                Console.WriteLine($"Owner: {subscription.Owner}");
                Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
                Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
                Console.WriteLine($"Endpoint: {subscription.Endpoint}");
                Console.WriteLine($"Protocol: {subscription.Protocol}");
                Console.WriteLine();
            }
        }
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 ListSubscriptions” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                 newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
                      << outcome.GetError().GetMessage()
                      <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
            for (auto const &subscription: subscriptions) {
```

```
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 ListSubscriptions” 中的。

CLI

AWS CLI

列出您的SNS订阅

以下`list-subscriptions`示例显示了您 AWS 账户中的SNS订阅列表。

```
aws sns list-subscriptions
```

输出：

```
{
    "Subscriptions": [
        {
            "Owner": "123456789012",
            "Endpoint": "my-email@example.com",
            "Protocol": "email",
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
            "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
        }
    ]
}
```

- 有关API详细信息，请参阅 “ListSubscriptions AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                    .build();

            ListSubscriptionsResponse result =
 snsClient.listSubscriptions(request);
```

```
            System.out.println(result.subscriptions());

        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 ListSubscriptions” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { ListSubscriptionsByTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to list
 subscriptions.
```

```
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 ListSubscriptions” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun listSNSSubscriptions() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- 有关API详细信息，请参阅ListSubscriptions中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 ListSubscriptions” 中的。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are
 returned.
        :return: An iterator that yields the subscriptions.
```

```
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
            logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- 有关API详细信息，请参阅ListSubscriptions中的 AWS SDKPython (Boto3) API 参考。

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# This class demonstrates how to list subscriptions to an Amazon Simple
 Notification Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
```

```
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = "SNS_TOPIC_ARN" # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for Ruby API参考 ListSubscriptions” 中的。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        oo_result = lo_sns->listsubscriptions( ).                " oo_result is
 returned for testing purposes. "
        DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
```

```
        MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list subscribers.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅ListSubscriptions中的AWS SDK以供SAPABAPAPI参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**ListTopics**一起使用 CLI

以下代码示例演示如何使用 `ListTopics`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// Lists the Amazon Simple Notification Service (Amazon SNS)
    /// topics for the current account.
    /// </summary>
    public class ListSNSTopics
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();
```

```
            await GetTopicListAsync(client);
        }

        /// <summary>
        /// Retrieves the list of Amazon SNS topics in groups of up to 100
        /// topics.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the list of topics.</param>
        public static async Task
GetTopicListAsync(IAmazonSimpleNotificationService client)
        {
            // If there are more than 100 Amazon SNS topics, the call to
            // ListTopicsAsync will return a value to pass to the
            // method to retrieve the next 100 (or less) topics.
            string nextToken = string.Empty;

            do
            {
                var response = await client.ListTopicsAsync(nextToken);
                DisplayTopicsList(response.Topics);
                nextToken = response.NextToken;
            }
            while (!string.IsNullOrEmpty(nextToken));
        }

        /// <summary>
        /// Displays the list of Amazon SNS Topic ARNs.
        /// </summary>
        /// <param name="topicList">The list of Topic ARNs.</param>
        public static void DisplayTopicsList(List<Topic> topicList)
        {
            foreach (var topic in topicList)
            {
                Console.WriteLine($"{topic.TopicArn}");
            }
        }
    }
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考 ListTopics” 中的。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage() <<
                      std::endl;
```

```
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考 ListTopics” 中的。

CLI

AWS CLI

列出你的SNS话题

以下`list-topics`示例列出了您 AWS 账户中的所有SNS主题。

```
aws sns list-topics
```

输出：

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- 有关API详细信息，请参阅 “ListTopics AWS CLI命令参考”。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
```

```
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Go API参考 ListTopics” 中的。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 ListTopics” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { ListTopicsCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const listTopics = async () => {
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 ListTopics” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun listSNSTopics() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- 有关API详细信息，请参阅ListTopics中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
```

```
 * Returns a list of the requester's topics from your AWS SNS account in the
 region specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 ListTopics” 中的。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
```

```
        """
        self.sns_resource = sns_resource


    def list_topics(self):
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- 有关API详细信息，请参阅ListTopics中的 AWS SDKPython (Boto3) API 参考。

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-sns"  # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
rescue StandardError => e
  puts "Error while listing the topics: #{e.message}"
  end
```

```
end

def run_me

  region = "REGION"
  sns_client = Aws::SNS::Resource.new(region: region)

  puts "Listing the topics."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for Ruby API参考 ListTopics” 中的。

Rust

SDK对于 Rust

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics() {
        println!("{}", topic.topic_arn().unwrap_or_default());
```

```
    }

    Ok(())
}
```

- 有关API详细信息，请参见ListTopics中的 Rust AWS SDK API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        oo_result = lo_sns->listtopics( ).            " oo_result is returned for
testing purposes. "
        DATA(lt_topics) = oo_result->get_topics( ).
        MESSAGE 'Retrieved list of topics.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list topics.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅ListTopics中的AWS SDK以供SAPABAPAPI参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**Publish**一起使用 CLI

以下代码示例演示如何使用 Publish。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建并发布到FIFO主题
- 发布一条SMS短信
- 将消息发布到队列

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

向主题发布消息。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
```

```
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

使用组、复制和属性选项向主题发布消息。

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
```

```
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }
```

```
                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
 messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
 false);
        }
    }
```

将用户的选择应用于发布操作。

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
```

```
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- 有关API详细信息，请参阅在AWS SDK for .NET API参考中发布。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

发布带有属性的消息。

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
```

```
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- 有关API详细信息，请参阅在AWS SDK for C++ API参考中发布。

CLI

AWS CLI

示例 1：向主题发布消息

以下publish示例将指定的消息发布到指定的SNS主题。该消息来自一个文本文件，您可以在该文件中包含换行符。

```
aws sns publish \
```

```
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
    --message file://message.txt
```

message.txt 的内容：

```
Hello World
Second Line
```

输出：

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

示例 2：向电话号码发布SMS消息

以下 publish 示例将消息 Hello world! 发布到电话号码 +1-555-555-0100。

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

输出：

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- 有关API详细信息，请参阅在《AWS CLI 命令参考》中发布。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- 有关API详细信息，请参阅在AWS SDK for Go API参考中发布。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅在AWS SDK for Java 2.x API参考中发布。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                  if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
```

```
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

使用组、复制和属性选项向主题发布消息。

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
```

```
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅在AWS SDK for JavaScript API参考中发布。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- 有关API详细信息，请参阅发布AWS SDK以获取 Kotlin API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅在AWS SDK for PHP API参考中发布。

PowerShell

用于 PowerShell

示例 1：此示例显示发布一条 MessageAttribute 声明为内联的消息。

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

示例 2：此示例显示发布一条事先 MessageAttributes 声明了多个消息的情况。

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- 有关API详细信息，请参阅在 AWS Tools for PowerShell Cmdlet 参考中发布。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

发布包含属性的消息，以便订阅可以根据属性进行筛选。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message.
 Values
                           must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
```

```
        else:
            return message_id
```

发布基于订阅者的协议采取不同形式的消息。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
        :param default_message: The default version of the message. This version
 is
                                sent to subscribers that have protocols that are
 not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
 subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
```

```
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
 MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

- 有关API详细信息，请参阅 Python 中发布AWS SDK以供参考 (Boto3) API。

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
```

```
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"         # Should be replaced with the actual message
 content

  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅在AWS SDK for Ruby API参考中发布。

Rust

SDK对于 Rust

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- 有关API详细信息，请参见发布AWSSDK以供 Rust API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
 testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message
        ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅发布AWSSDK以供SAPABAPAPI参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**SetSMSAttributes**一起使用 CLI

以下代码示例演示如何使用 `SetSMSAttributes`。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

如何使用 Amazon SNS 来设置 D efaultSMSType 属性。

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for C++ API参考资料》中的 S。

CLI

AWS CLI

设置SMS消息属性

以下set-sms-attributes示例将SMS邮件的默认发件人 ID 设置为MyName。

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

此命令不生成任何输出。

- 有关API详细信息，请参阅etSMSAttributes《AWS CLI 命令参考》中的 S。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
```

```
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for Java 2.x API参考资料》中的 S。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional – (Default) Noncritical messages, such as marketing
 messages.
        // Transactional – Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for JavaScript API参考资料》中的S。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅etSMSAttributes《AWS SDK for PHP API参考资料》中的 S。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**SetSubscriptionAttributes**一起使用 CLI

以下代码示例演示如何使用 SetSubscriptionAttributes。

CLI

AWS CLI

设置订阅属性

以下set-subscription-attributes示例将该RawMessageDelivery属性设置为订SQS阅。

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

此命令不生成任何输出。

以下set-subscription-attributes示例为SQS订阅设置一个FilterPolicy属性。

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

此命令不生成任何输出。

以下set-subscription-attributes示例从SQS订阅中删除了该FilterPolicy属性。

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

此命令不生成任何输出。

- 有关API详细信息，请参阅 “SetSubscriptionAttributes AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 SetSubscriptionAttributes” 中的。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must
 have an
        attribute that passes the filter or it will not be sent to the
 subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
 filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
 AttributeValue=json.dumps(att_policy)
```

```
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- 有关API详细信息，请参阅SetSubscriptionAttributes中的 AWS SDKPython (Boto3) API 参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或SetSubscriptionAttributesRedrivePolicy一起使用 CLI

以下代码示例显示了如何使用SetSubscriptionAttributesRedrivePolicy。

Java

SDK适用于 Java 1.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
```

```
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**SetTopicAttributes**一起使用 CLI

以下代码示例演示如何使用 SetTopicAttributes。

CLI

AWS CLI

为主题设置属性

以下 set-topic-attributes 示例为指定主题设置 DisplayName 属性。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

此命令不生成任何输出。

- 有关API详细信息，请参阅 “SetTopicAttributes AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
```

```
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
            SetTopicAttributesRequest request =
 SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 SetTopicAttributes” 中的。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考 SetTopicAttributes” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- 有关API详细信息，请参阅SetTopicAttributes中的 Kotlin AWS SDK API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅 “AWS SDK for PHP API参考 SetTopicAttributes” 中的。

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: "2008-10-17",
      Id: "__default_policy_ID",
      Statement: [{
                   Sid: "__default_statement_ID",
                   Effect: "Allow",
                   Principal: { "AWS": "*" },
                   Action: ["SNS:Publish"],
                   Resource: topic_arn,
                   Condition: {
                     ArnEquals: {
                       "AWS:SourceArn": resource_arn
                     }
                   }
                 }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN"     # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME"    # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for Ruby API参考 SetTopicAttributes” 中的。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
TRY.
    lo_sns->settopicattributes(
        iv_topicarn = iv_topic_arn
        iv_attributename  = iv_attribute_name
        iv_attributevalue = iv_attribute_value
    ).
    MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关API详细信息，请参阅SetTopicAttributes中的AWS SDK以供SAPABAPAPI参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**Subscribe**一起使用 CLI

以下代码示例演示如何使用 `Subscribe`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建并发布到FIFO主题
- 将消息发布到队列

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
/// <summary>
/// Creates a new subscription to a topic.
/// </summary>
/// <param name="client">The initialized Amazon SNS client object, used
/// to create an Amazon SNS subscription.</param>
/// <param name="topicArn">The ARN of the topic to subscribe to.</param>
/// <returns>A SubscribeResponse object which includes the subscription
/// ARN for the new subscription.</returns>
public static async Task<SubscribeResponse> TopicSubscribeAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    SubscribeRequest request = new SubscribeRequest()
    {
        TopicArn = topicArn,
        ReturnSubscriptionArn = true,
        Protocol = "email",
        Endpoint = "recipient@example.com",
    };

    var response = await client.SubscribeAsync(request);

    return response;
}
```

使用可选筛选条件为队列订阅主题。

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- 有关API详细信息，请参阅AWS SDK for .NET API参考资料中的订阅。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

为移动应用程序订阅主题。

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

为主题订阅 Lambda 函数。

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

为SQS队列订阅主题。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
   Aws::SNS::SNSClient snsClient(clientConfiguration);

           Aws::SNS::Model::SubscribeRequest request;
           request.SetTopicArn(topicARN);
           request.SetProtocol("sqs");
           request.SetEndpoint(queueARN);

           Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

           if (outcome.IsSuccess()) {
               Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
               std::cout << "The queue '" << queueName
                         << "' has been subscribed to the topic '"
                         << "'" << topicName << "'" << std::endl;
               std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                         << std::endl;
               subscriptionARNS.push_back(subscriptionARN);
           }
           else {
               std::cerr << "Error with TopicsAndQueues::Subscribe. "
                         << outcome.GetError().GetMessage()
                         << std::endl;

               cleanUp(topicARN,
                       queueURLS,
                       subscriptionARNS,
                       snsClient,
                       sqsClient);

               return false;
           }
```

使用过滤器订阅主题。

```
       static const Aws::String TONE_ATTRIBUTE("tone");
       static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                                      "sincere"};
```

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;
```

```
                    request.AddAttributes("FilterPolicy", jsonPolicy);
                }
                else {
                    std::cout
                            << "Because you did not select any attributes, no
 filter "
                            << "will be added to this subscription." <<
 std::endl;
                }
            }
        }  // if (isFifoTopic)
        Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

        if (outcome.IsSuccess()) {
            Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
            std::cout << "The queue '" << queueName
                      << "' has been subscribed to the topic '"
                      << "'" << topicName << "'" << std::endl;
            std::cout << "with the subscription ARN '" << subscriptionARN <<
 "."
                      << std::endl;
            subscriptionARNS.push_back(subscriptionARN);
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Subscribe. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
```

```
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";
```

```
        result = jsonPolicyStream.str();
    }

    return result;
}
```

- 有关API详细信息，请参阅AWS SDK for C++ API参考资料中的订阅。

CLI

AWS CLI

订阅主题

以下 subscribe 命令将电子邮件地址订阅到指定主题。

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

输出：

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- 有关API详细信息，请参阅《AWS CLI 命令参考》中的 “订阅”。

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用可选筛选条件为队列订阅主题。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
```

```
}
```

- 有关API详细信息，请参阅AWS SDK for Go API参考资料中的订阅。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
 email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

为HTTP终端节点订阅主题。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive
 notifications.
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
```

```
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn()
 + "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

为主题订阅 Lambda 函数。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>
```

```
                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考资料中的订阅。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
```

```
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

为移动应用程序订阅主题。

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
 *                            when an application registers for notifications.
 */
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
```

```
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

为主题订阅 Lambda 函数。

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
```

```
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

为SQS队列订阅主题。

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

使用过滤器订阅主题。

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。

- 有关API详细信息，请参阅AWS SDK for JavaScript API参考资料中的订阅。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

为主题订阅 Lambda 函数。

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
```

```
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- 有关API详细信息，请参阅订阅AWS SDK以获取 Kotlin API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */
```

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

为HTTP终端节点订阅主题。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
```

```
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关API详细信息，请参阅AWS SDK for PHP API参考资料中的订阅。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
```

```
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
number
                         (in E.164 format) for SMS messages, or an email address
for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
topic.arn
            )
            raise
        else:
            return subscription
```

- 有关API详细信息，请参阅订阅以获取 Python (Boto3) API 参考。AWS SDK

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info("Subscription created successfully.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
```

```
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
```

- 有关更多信息，请参阅 AWS SDK for Ruby 开发人员指南。
- 有关API详细信息，请参阅AWS SDK for Ruby API参考资料中的订阅。

Rust

SDK对于 Rust

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);
```

```
    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- 有关API详细信息，请参阅订阅AWS SDK以获取 Rust API 参考。

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过电子邮件地址订阅主题。

```
    TRY.
        oo_result = lo_sns->subscribe(                      "oo_result is
returned for testing purposes."
                iv_topicarn = iv_topic_arn
                iv_protocol = 'email'
```

```
                iv_endpoint = iv_email_address
                iv_returnsubscriptionarn = abap_true
            ).
        MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅订阅AWSSDK以供SAPABAPAPI参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**TagResource**一起使用 CLI

以下代码示例演示如何使用 TagResource。

CLI

AWS CLI

为主题添加标签

以下tag-resource示例向指定的 Amazon SNS 主题添加元数据标签。

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

此命令不生成任何输出。

- 有关API详细信息，请参阅 “TagResource AWS CLI命令参考”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考 TagResource” 中的。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

    val tag2 =
        Tag {
            key = "Environment"
            value = "Gamma"
        }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request =
        TagResourceRequest {
            resourceArn = topicArn
            tags = tagList
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- 有关API详细信息，请参阅TagResource中的 Kotlin AWS SDK API 参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

与 AWS SDK或**Unsubscribe**一起使用 CLI

以下代码示例演示如何使用 Unsubscribe。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 将消息发布到队列

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

通过订阅取消订阅ARN某个主题。

```
/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- 有关API详细信息，请参阅 “AWS SDK for .NET API参考资料” 中的 “取消订阅”。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅 “AWS SDK for C++ API参考资料” 中的 “取消订阅”。

CLI

AWS CLI

从主题取消订阅

以下 unsubscribe 示例将从主题删除指定的订阅。

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

此命令不生成任何输出。

- 有关API详细信息，请参阅《AWS CLI 命令参考》中的 “取消订阅”。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- 有关API详细信息，请参阅 “AWS SDK for Java 2.x API参考资料” 中的 “取消订阅”。

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

在单独的模块中创建客户端并将其导出。

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

导入SDK和客户端模块并调用API。

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
```

```
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for JavaScript API参考资料” 中的 “取消订阅”。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- 有关API详细信息，请参阅 Kotlin API 参考中的AWS SDK取消订阅。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅 “AWS SDK for PHP API参考资料” 中的 “取消订阅”。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

- 有关API详细信息，请参阅 Python (Boto3) 参考中的AWS SDK取消订阅。API

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅中的取消订阅AWSSDK以供SAPABAPAPI参考。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

Amazon SNS 使用的场景 AWS SDKs

以下代码示例向您展示了如何使用在 Amazon SNS 中实现常见场景 AWS SDKs。这些场景向您展示了如何通过调用 Amazon 中的多个函数SNS或与其他函数结合使用来完成特定任务 AWS 服务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 构建应用程序以将数据提交到 DynamoDB 表
- 构建转换消息的发布和订阅应用程序
- 使用 Amazon SNS 推送通知创建平台终端节点 AWS SDK
- 创建照片资产管理应用程序，让用户能够使用标签管理照片
- 创建 Amazon Textract 浏览器应用程序
- 使用创建并发布到 A FIFO mazon SNS 主题 AWS SDK
- 使用 Amazon Rekognition 检测视频中的人物和物体 AWS SDK
- 使用SMS向 Amazon SNS 主题发布消息 AWS SDK
- 使用 Amazon S3 SNS 向亚马逊发布一条大消息 AWS SDK
- 使用发布一条 Amazon SNS SMS 短信 AWS SDK
- 使用以下命令将亚马逊SNS消息发布到亚马逊SQS队列 AWS SDK
- 使用API网关调用 Lambda 函数
- 使用计划的事件调用 Lambda 函数

构建应用程序以将数据提交到 DynamoDB 表

以下代码示例展示如何构建将数据提交到 Amazon DynamoDB 表并在用户更新该表时通知您的应用程序。

Java

SDK适用于 Java 2.x

演示如何创建动态 Web 应用程序，该应用程序使用 Amazon DynamoDB API Java 提交数据并使用亚马逊简单通知服务 Java 发送短信。API

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB
- Amazon SNS

JavaScript

SDK对于 JavaScript (v3)

此示例说明如何构建一个应用程序，让用户能够使用亚马逊简单通知服务 (Amazon) 向 Amazon DynamoDB 表提交数据，以及如何使用亚马逊简单通知服务 (Amazon) 向管理员发送短信。SNS

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

该示例也可在 AWS SDK for JavaScript v3 开发人员指南中找到。

本示例中使用的服务

- DynamoDB
- Amazon SNS

Kotlin

SDK对于 Kotlin 来说

演示如何创建原生安卓应用程序，该应用程序使用亚马逊 DynamoDB API Kotlin 提交数据并使用亚马逊 Kotlin 发送短信。SNS API

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB
- Amazon SNS

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

构建转换消息的发布和订阅应用程序

以下代码示例展示如何创建具有订阅和发布功能并能转换消息的应用程序。

.NET

AWS SDK for .NET

演示如何使用 Amazon 简单通知服务。 NETAPI创建具有订阅和发布功能的 Web 应用程序。此外，此示例应用程序还会转换消息。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon SNS
- Amazon Translate

Java

SDK适用于 Java 2.x

演示如何使用 Amazon 简单通知服务 Java API 创建具有订阅和发布功能的 Web 应用程序。此外，此示例应用程序还会转换消息。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

有关如何设置和运行使用 Java Async 的示例的完整源代码和说明API，请参阅上的GitHub完整示例。

本示例中使用的服务

- Amazon SNS
- Amazon Translate

Kotlin

SDK对于 Kotlin 来说

演示如何使用 Amazon SNS Kotlin API 创建具有订阅和发布功能的应用程序。此外，此示例应用程序还会转换消息。

有关如何创建 Web 应用程序的完整源代码和说明，请参阅上的完整示例GitHub。

有关如何创建原生 Android 应用程序的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon SNS
- Amazon Translate

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用 Amazon SNS 推送通知创建平台终端节点 AWS SDK

以下代码示例展示了如何为 Amazon SNS 推送通知创建平台终端节点。

CLI

AWS CLI

创建平台应用程序端点

以下 `create-platform-endpoint` 示例使用指定令牌为指定平台应用程序创建端点。

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

输出：

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

创建照片资产管理应用程序，让用户能够使用标签管理照片

以下代码示例演示了如何创建无服务器应用程序，让用户能够使用标签管理照片。

.NET

AWS SDK for .NET

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API网关
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK对于 C++

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API网关
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK适用于 Java 2.x

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API网关
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK对于 JavaScript (v3)

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API网关
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK对于 Kotlin 来说

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API网关
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK for PHP

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API网关
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK对于 Rust

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 GitHub。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API网关
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

创建 Amazon Textract 浏览器应用程序

以下代码示例展示如何通过交互式应用程序探索 Amazon Textract 输出。

JavaScript

SDK对于 JavaScript (v3)

演示如何使用 AWS SDK for JavaScript 来构建 React 应用程序，该应用程序使用 Amazon Textract 从文档图像中提取数据并将其显示在交互式网页中。此示例在 Web 浏览器中运行，需要经过身份验证的 Amazon Cognito 身份才能获得凭证。它使用亚马逊简单存储服务 (Amazon S3) Service 进行存储，对于通知，它会轮询订阅亚马逊简单通知服务 (A SQS mazon) 主题的亚马逊简单队列服务 (Amazon) 队列。SNS

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK适用于 Python (Boto3)

演示如何 AWS SDK for Python (Boto3) 与 Amazon Textract 配合使用来检测文档图像中的文本、表单和表格元素。输入图像和 Amazon Textract 输出在 Tkinter 应用程序中显示，该应用程序可让您探索检测到的元素。

- 将文档图像提交到 Amazon Textract 并探索检测到的元素的输出。
- 将图像直接提交到 Amazon Textract，或通过 Amazon Simple Storage Service（Amazon S3）桶提交图像。
- 使用异步APIs启动任务，该任务在任务完成时向亚马逊简单通知服务 (AmazonSNS) 主题发布通知。
- 轮询亚马逊简单队列服务 (AmazonSQS) 队列以获取任务完成消息并显示结果。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用创建并发布到 A FIFO mazon SNS 主题 AWS SDK

以下代码示例展示了如何创建和发布到 A FIFO mazon SNS 主题。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

此示例

- 创建一个 Amazon SNS FIFO 主题、两个亚马逊SQSFIFO队列和一个标准队列。
- 将队列订阅到主题，发布一条消息到主题。

该测试验证每个队列是否收到消息。完整的示例还显示了添加访问策略，并在最后删除了资源。

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
 created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
 will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
 ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));
```

```
        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                    .topicArn(topicARN)
                    .endpoint(queue.queueARN)
                    .protocol("sqs")
                    .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(attributeValue)
                    .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                    .topicArn(topicArn)
                    .subject(subject)
```

```
                    .message(payload)
                    .messageGroupId(groupId)
                    .messageDeduplicationId(dedupId)
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关API详细信息，请参阅 “参AWS SDK for Java 2.x API考” 中的以下主题。
  - CreateTopic
  - Publish
  - 订阅

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建亚马逊SNSFIFO主题，为该主题订阅亚马逊SQSFIFO和标准队列，然后向该主题发布消息。

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
```

```
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")
```

```
    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
```

```
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
```

```
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error
```

```
    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
```

```
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- 有关API详细信息，请参阅 Python (Boto3) API 参考中的AWS SDK以下主题。
  - CreateTopic
  - Publish
  - 订阅

SAP ABAP

SDK对于 SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建FIFO主题，在 Amazon SQS FIFO 队列中订阅该主题，然后向亚马逊SNS主题发布消息。

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
```

```
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                        "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn
           ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                          "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
```

```
            iv_message = 'The price of your mobile plan has been increased from
$19 to $23'
            iv_subject = 'Changes to mobile plan'
            iv_messagegroupid = 'Update-2'
            iv_messagededuplicationid = 'Update-2.1'
            it_messageattributes = lt_msg_attributes
      ).
        ov_message_id = lo_result->get_messageid( ).                    "
ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关API详细信息，请参阅中的以下主题AWS SDK以供SAPABAPAPI参考。
  - CreateTopic
  - Publish
  - 订阅

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用 Amazon Rekognition 检测视频中的人物和物体 AWS SDK

以下代码示例展示如何使用 Amazon Rekognition 检测视频中的人物和对象。

Python

SDK适用于 Python (Boto3)

通过启动异步检测任务，使用 Amazon Rekognition 来检测视频中的人脸、对象和人物。此示例还将 Amazon Rekognition 配置为在任务完成并订阅亚马逊简单队列服务 SNS (Amazon) 队列时通知亚马逊简单通知服务 SQS (Amazon) 主题。当队列收到有关任务的消息时，将检索该任务并输出结果。

最好在上查看此示例 GitHub。有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用SMS向 Amazon SNS 主题发布消息 AWS SDK

以下代码示例展示了如何：

- 创建 Amazon SNS 主题。
- 使用手机号码订阅主题。
- 向主题发布SMS消息，以便所有订阅的电话号码都能同时收到消息。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建一个主题并返回它ARN。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return "";
    }
}
```

为终端节点订阅主题。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives
 notifications (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

设置消息的属性，例如发件人的 ID、最高价格及其类型。消息属性是可选的。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

向主题发布消息。消息将会发送到每个订阅者。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
```

```
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用 Amazon S3 SNS 向亚马逊发布一条大消息 AWS SDK

以下代码示例演示如何SNS使用 Amazon S3 向亚马逊发布大消息来存储消息负载。

Java

SDK适用于 Java 1.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

要发布大型消息，请使用适用于 Java 的 Amazon SNS 扩展客户端库。您发送的消息将引用包含实际消息内容的 Amazon S3 对象。

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {

        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;

                // Message threshold controls the maximum message size that will
 be allowed to
                // be published
                // through SNS using the extended client. Payload of messages
 exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
 maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
                final AmazonSNS snsClient =
 AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
 AmazonSQSClientBuilder.standard().withRegion(region).build();
                final AmazonS3 s3Client =
 AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
```

```
                final String topicArn = snsClient.createTopic(
                                new
CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);

                // To read message content stored in S3 transparently through SQS
extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
                final SetSubscriptionAttributesRequest
subscriptionAttributesRequest = new SetSubscriptionAttributesRequest();

subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");

snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);

                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3
when the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
                final SNSExtendedClientConfiguration
snsExtendedClientConfiguration = new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

.withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);

                // Publish message via SNS with storage in S3
                final String message = "This message is stored in S3 as it
exceeds the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
```

```
                final ExtendedClientConfiguration sqsExtendedClientConfiguration
 = new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client,
 BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
 AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
 sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
 result.getMessages().get(0).getBody());
        }
}
```

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用发布一条 Amazon SNS SMS 短信 AWS SDK

以下代码示例展示了如何使用 Amazon 发布SMS消息SNS。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;
```

```
    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
```

```
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- 有关API详细信息，请参阅在AWS SDK for .NET API参考中发布。

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
```

```
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关API详细信息，请参阅在AWS SDK for C++ API参考中发布。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅在AWS SDK for Java 2.x API参考中发布。

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- 有关API详细信息，请参阅发布AWS SDK以获取 Kotlin API 参考。

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
```

```
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 有关更多信息，请参阅 AWS SDK for PHP 开发人员指南。
- 有关API详细信息，请参阅在AWS SDK for PHP API参考中发布。

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- 有关API详细信息，请参阅 Python 中发布AWS SDK以供参考 (Boto3) API。

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用以下命令将亚马逊SNS消息发布到亚马逊SQS队列 AWS SDK

以下代码示例演示了如何：

- 创建主题（FIFO或非-FIFO）。
- 针对主题订阅多个队列，并提供应用筛选条件的选项。
- 将消息发布到主题。
- 轮询队列中是否有收到的消息。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

在命令提示符中运行交互式场景。

```
/// <summary>
/// Console application to run a workflow scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
```

```
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
```

```
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues workflow is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the workflow.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this workflow, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
```

```
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
_useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;
```

```
                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must
be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
queueName)
    {
        if (_useFifoTopic)
        {
```

```
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
```

```
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
```

```
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }
```

```
                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
```

```
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
```

```
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
```

```
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

创建一个封装 Amazon SQS 操作的类。

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;

    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
 useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
```

```
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
```

```
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
                                                $"\"aws:SourceArn\":
\"{topicArn}\"" +
                                                "}" +
                                           "}" +
                                "}]" +
                              "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
```

```
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
 List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
 _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

创建一个封装 Amazon SNS 操作的类。

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
```

```
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
```

```
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
 { { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
```

```
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
_amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
```

```
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- 有关API详细信息，请参阅 “参AWS SDK for .NET API考” 中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - 订阅
  - Unsubscribe

C++

SDK对于 C++

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
```

```
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
 the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the
 queues."
              << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication
 and message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
 supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or
 automatically generated "
                << "from content using a hash function." << std::endl;
        std::cout
```

```
                << "If a message is successfully published to an SNS FIFO topic,
 any message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
                << "within the five-minute deduplication interval, is accepted
 but not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name."
                    << std::endl;
        }
```

```
        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());
```

```
            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                      "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

```
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
```

```
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
```

```
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";
```

```
                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
```

```
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
```

```
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);
```

```
    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

```
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
```

```
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

        Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
```

```
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                          const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
```

```
}
```

- 有关API详细信息，请参阅 “参AWS SDK for C++ API考” 中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - 订阅
  - Unsubscribe

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

在命令提示符中运行交互式场景。

```
const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
```

```
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic() (string, string, bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
```

```
    "the topic name.", FIFO_SUFFIX)
  }

  topicArn, err := runner.snsActor.CreateTopic(topicName, isFifoTopic,
  contentBasedDeduplication)
  if err != nil {
   panic(err)
  }
  log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
  \n"+
   "'%v' has been created.", topicName, topicArn)

  return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ordinal string, isFifoTopic bool)
 (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 queueName string, queueUrl string, topicName string, topicArn string, ordinal
 string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(queueUrl)
 if err != nil {
  panic(err)
```

```
}
log.Printf("The ARN of your queue is: %v.\n", queueArn)

err = runner.sqsActor.AttachSendMessagePolicy(queueUrl, queueArn, topicArn)
if err != nil {
 panic(err)
}
log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
 "messages to it.")
log.Println(strings.Repeat("-", 88))

var filterPolicy map[string][]string
if isFifoTopic {
 if ordinal == "first" {
  log.Println("Subscriptions to a FIFO topic can have filters.\n" +
   "If you add a filter to this subscription, then only the filtered messages\n"
+
   "will be received in the queue.\n" +
   "For information about message filtering, see\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
   "For this example, you can filter messages by a \"tone\" attribute.")
 }

 wantFiltering := runner.questioner.AskBool(
  fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
   "from the %v topic? (y/n) ", queueName, topicName), "y")
 if wantFiltering {
  log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

  var toneSelections []string
  askAboutTones := true
  for askAboutTones {
   toneIndex := runner.questioner.AskChoice(
    "Enter the number of the tone you want to filter by:\n", ToneChoices)
   toneSelections = append(toneSelections, ToneChoices[toneIndex])
   askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
  }
  log.Printf("Your subscription will be filtered to only pass the following
tones: %v\n", toneSelections)
  filterPolicy = map[string][]string{TONE_KEY: toneSelections}
 }
}
```

```
 subscriptionArn, err := runner.snsActor.SubscribeQueue(topicArn, queueArn,
 filterPolicy)
 if err != nil {
  panic(err)
 }
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(topicArn string, isFifoTopic bool,
 contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
```

```
  }

  err := runner.snsActor.Publish(topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(queueUrls []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages),
 queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
```

```
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(queueUrl, messages)
   if err != nil {
    panic(err)
   }
  }
 }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup()
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this workflow, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
```

```
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic()
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
```

```
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup()
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

定义一个封装本示例中使用的 Amazon SNS 操作的结构。

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
```

```
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
```

```
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
```

```
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

定义一个封装本示例中使用的 Amazon SQS 操作的结构。

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(queueName string, isFifoQueue bool) (string,
 error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(context.TODO(), &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}
```

```
// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(queueUrl string) (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(context.TODO(),
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(queueUrl string, queueArn string,
 topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
```

```
 }
 _, err = actor.SqsClient.SetQueueAttributes(context.TODO(),
 &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(queueUrl string, maxMessages int32, waitTime
 int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(context.TODO(),
 &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
```

```
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(queueUrl string, messages []types.Message)
 error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(context.TODO(),
 &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(context.TODO(), &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
```

```
 }
 return err
}
```

- 有关API详细信息，请参阅 “参AWS SDK for Go API考” 中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - 订阅
  - Unsubscribe

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
```

```
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 *
 * This Java example performs these tasks:
 *
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
                "Usage:\n" +
                "    <fifoQueueARN>\n\n" +
                "Where:\n" +
                "    accountId - Your AWS account Id value.";

        // if (args.length != 1) {
        // System.out.println(usage);
        // System.exit(1);
        // }

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)

 .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_EAST_1)

 .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();

        Scanner in = new Scanner(System.in);
```

```
        String accountId = "814548047983";
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this workflow, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
                "You can select from several options for configuring the topic
 and the subscriptions for the queue.\n" +
                "You can then post to the topic and see the results in the
 queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
                "FIFO topics deliver messages in order and support deduplication
 and message filtering.\n" +
                "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
 deduplication is supported.\n" +
                    "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                    +
```

```
                    "        If a message is successfully published to an SNS
FIFO topic, any message published and determined to have the same deduplication
ID,\n"
                    +
                    "        within the five-minute deduplication interval, is
accepted but not delivered.\n" +
                    "        For more information about deduplication, see
https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                    "Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo'
must be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = "{\n" +
                "     \"Statement\": [\n" +
                "     {\n" +
                "         \"Effect\": \"Allow\",\n" +
                "                 \"Principal\": {\n" +
                "             \"Service\": \"sns.amazonaws.com\"\n" +
                "         },\n" +
                "         \"Action\": \"sqs:SendMessage\",\n" +
                "                 \"Resource\": \"arn:aws:sqs:us-east-1:" +
accountId + ":" + sqsQueueName + "\",\n" +
                "                 \"Condition\": {\n" +
                "             \"ArnEquals\": {\n" +
                "                 \"aws:SourceArn\": \"arn:aws:sns:us-east-1:" +
accountId + ":" + topicName + "\"\n" +
                "             }\n" +
                "         }\n" +
                "     }\n" +
                "     ]\n" +
                " }";
```

```
        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                    "If you add a filter to this subscription, then only the
filtered messages will be received in the queue.\n"
                            +
                            "For information about message filtering, see
https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                            +
                            "For this example, you can filter messages by a
\"tone\" attribute.");
            System.out.println("Would you like to filter messages for " +
sqsQueueName + "'s subscription to the topic "
                    + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
```

```
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this
message?  (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
```

```
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
duplication, groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }
```

```
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
```

```
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
                    + "\nSubscription was removed for " +
request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                        .id(msg.messageId())
                        .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
```

```
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .maxNumberOfMessages(5)
                        .build();
                return
sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .messageAttributeNames(msgAttValue) // Include other
message attributes if needed.
                        .maxNumberOfMessages(5)
                        .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    public static void pubMessageFIFO(SnsClient snsClient,
            String message,
            String topicArn,
            String msgAttValue,
            String duplication,
            String groupId,
            String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                } else {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
                        .dataType("String")
                        .stringValue("true")
                        .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
```

```
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                            .topicArn(topicArn)
                            .message(message)
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .messageAttributes(messageAttributes)
                            .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
                        .topicArn(topicArn)
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
```

```
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
                        .topicArn(topicArn)
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                        .subscriptionArn(result.subscriptionArn())
                        .attributeName(attributeName)
                        .attributeValue(updatedJsonString)
                        .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
```

```
   public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
       try {
           Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
           attrMap.put(QueueAttributeName.POLICY, policy);

           SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                   .queueUrl(queueUrl)
                   .attributes(attrMap)
                   .build();

           sqsClient.setQueueAttributes(attributesRequest);
           System.out.println("The policy has been successfully attached.");

       } catch (SnsException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
       // Specify the attributes to retrieve.
       List<QueueAttributeName> atts = new ArrayList<>();
       atts.add(QueueAttributeName.QUEUE_ARN);

       GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
               .queueUrl(queueUrl)
               .attributeNames(atts)
               .build();

       GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
       Map<String, String> queueAtts = response.attributesAsStrings();
       for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
           return queueAtt.getValue();

       return "";
   }

   public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
```

```
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                        .queueName(queueName)
                        .attributes(attrs)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                        .queueName(queueName)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();
```

```
            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关API详细信息，请参阅 “参AWS SDK for Java 2.x API考” 中的以下主题。
  - CreateQueue

- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Publish
- ReceiveMessage
- SetQueueAttributes
- 订阅
- Unsubscribe

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

这是此工作流程的入口点。

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { SlowLogger } from "@aws-doc-sdk-examples/lib/slow-logger.js";

export const startSnsWorkflow = () => {
  const noLoggerDelay = process.argv.find((arg) => arg === "--no-logger-delay");
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = noLoggerDelay ? console : new SlowLogger(25);

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);
```

```
  wkflw.start();
};
```

前面的代码提供必要的依赖关系并启动工作流程。下一节包含示例的大部分内容。

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
   */
  constructor(snsClient, sqsClient, prompter, logger) {
    this.snsClient = snsClient;
```

```
    this.sqsClient = sqsClient;
    this.prompter = prompter;
    this.logger = logger;
  }

  async welcome() {
    await this.logger.log(MESSAGES.description);
  }

  async confirmFifo() {
    await this.logger.log(MESSAGES.snsFifoDescription);
    this.isFifo = await this.prompter.confirm({
      message: MESSAGES.snsFifoPrompt,
    });

    if (this.isFifo) {
      this.logger.logSeparator(MESSAGES.headerDedup);
      await this.logger.log(MESSAGES.deduplicationNotice);
      await this.logger.log(MESSAGES.deduplicationDescription);
      this.autoDedup = await this.prompter.confirm({
        message: MESSAGES.deduplicationPrompt,
      });
    }
  }

  async createTopic() {
    await this.logger.log(MESSAGES.creatingTopics);
    this.topicName = await this.prompter.input({
      message: MESSAGES.topicNamePrompt,
    });
    if (this.isFifo) {
      this.topicName += ".fifo";
      this.logger.logSeparator(MESSAGES.headerFifoNaming);
      await this.logger.log(MESSAGES.appendFifoNotice);
    }

    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
```

```
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    let maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );
```

```
      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
```

```
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
```

```
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
```

```
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
```

```
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
```

```
    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });

    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
```

```
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- 有关API详细信息，请参阅 “参AWS SDK for JavaScript API考” 中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - 订阅
  - Unsubscribe

Kotlin

SDK对于 Kotlin 来说

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
```

```
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
 queues.")
    println(
        """
                In this workflow, you will create an SNS topic and subscribe an
 SQS queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication
 and message filtering.
                Would you like to work with FIFO topics? (y/n)
```

```
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
 generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
```

```
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
```

```
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/
latest/dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's
 subscription to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the
 following \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
```

```
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following
\"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
```

```
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key
 to continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
```

```
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
```

```
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }
```

```
            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
```

```
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
 + topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
```

```
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
```

```
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }
```

```
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
```

```
}
```

- 有关API详细信息，请参阅中的以下主题以获取 Kotlin AWS SDK API 参考。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - 订阅
  - Unsubscribe

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用API网关调用 Lambda 函数

以下代码示例展示了如何创建由 Amazon API Gateway 调用的 AWS Lambda 函数。

Java

SDK适用于 Java 2.x

演示如何使用 Lambda Java 运行时创建 AWS Lambda 函数。API此示例调用不同的 AWS 服务来执行特定的用例。此示例演示如何创建由 Amazon API Gateway 调用的 Lambda 函数，该函数会扫描亚马逊 DynamoDB 表中的工作周年纪念日，并使用亚马逊简单通知服务 (Amaz SNS on) 向您的员工发送一条短信，祝贺他们在一周年之日到来。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- API网关

- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK对于 JavaScript (v3)

演示如何使用 Lambda JavaScript 运行时创建 AWS Lambda 函数。API此示例调用不同的 AWS 服务来执行特定的用例。此示例演示如何创建由 Amazon API Gateway 调用的 Lambda 函数，该函数会扫描亚马逊 DynamoDB 表中的工作周年纪念日，并使用亚马逊简单通知服务 (Amaz SNS on) 向您的员工发送一条短信，祝贺他们在一周年之日到来。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

该示例也可在 AWS SDK for JavaScript v3 开发人员指南中找到。

本示例中使用的服务

- API网关
- DynamoDB
- Lambda
- Amazon SNS

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用计划的事件调用 Lambda 函数

以下代码示例说明如何创建由 Amazon EventBridge 计划事件调用的 AWS Lambda 函数。

Java

SDK适用于 Java 2.x

演示如何创建调用函数的 Amazon EventBridge 计划事件。 AWS Lambda 配置 EventBridge 为使用 cron 表达式来调度 Lambda 函数的调用时间。在此示例中，您将使用 Lambda Java 运行时创建一个 Lambda 函数。API此示例调用不同的 AWS 服务来执行特定的用例。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK对于 JavaScript (v3)

演示如何创建调用函数的 Amazon EventBridge 计划事件。 AWS Lambda 配置 EventBridge 为使用 cron 表达式来调度 Lambda 函数的调用时间。在此示例中，您将使用 Lambda 运行时创建一个 Lambda 函数。 JavaScript API此示例调用不同的 AWS 服务来执行特定的用例。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

该示例也可在 AWS SDK for JavaScript v3 开发人员指南中找到。

本示例中使用的服务

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

使用 Amazon SNS 的无服务器示例 AWS SDKs

以下代码示例展示了如何将 Amazon SNS 与配合使用 AWS SDKs。

示例

从亚马逊触发器调用 Lambda 函数 SNS

以下代码示例说明如何实现一个 Lambda 函数，该函数接收通过接收来自主题的消息而触发的事件。SNS该函数从事件参数检索消息并记录每条消息的内容。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用使用 Lambda 使用一个SNS事件。 NET。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
```

```
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK适用于 Go V2

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Go 在 Lambda 上使用SNS事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
```

```
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 在 Lambda 上使用SNS事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

JavaScript

SDK对于 JavaScript (v3)

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用使用 Lambda 使用一个SNS事件。 JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

使用使用 Lambda 使用一个SNS事件。 TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
```

```
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK for PHP

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用使用 Lambda 使用一个SNS事件。PHP

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
```

```
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-
lambda-custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be
 marked as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Python

SDK适用于 Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Python 在 Lambda 中使用SNS事件。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK对于 Ruby

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Ruby 在 Lambda 上使用SNS事件。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
```

```
  raise
end
```

Rust

SDK对于 Rust

> **Note**
>
> 还有更多相关信息 GitHub。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Rust 在 Lambda 上使用SNS事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features
 = ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);
```

```
    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

有关 AWS SDK开发者指南和代码示例的完整列表，请参阅将 Amazon SNS 与 AWS SDK。本主题还包括有关入门的信息以及有关先前SDK版本的详细信息。

Amazon SNS 安全

本节提供有关亚马逊SNS安全、身份验证和访问控制以及亚马逊SNS访问策略语言的信息。

主题

亚马逊SNS数据保护

分 AWS 担责任模型适用于亚马逊简单通知服务中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础架构上的内容的控制。此内容包括您使用的 AWS 服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私FAQ。 有关欧洲数据保护的信息，请参阅责任AWS 共担模型和AWS安全GDPR博客上的博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭据并使用 AWS Identity and Access Management (IAM) 设置个人用户帐户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用SSL/TLS与 AWS 资源通信。我们推荐 TLS 1.2 或更高版本。
- 使用API进行设置和用户活动记录 AWS CloudTrail。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的个人数据。
- 如果您在 AWS 通过命令行界面或访问时需要 FIPS 140-2 经过验证的加密模块API，请使用端点。FIPS有关可用FIPS端点的更多信息，请参阅联邦信息处理标准 (FIPS) 140-2。
- 消息数据保护

- 消息数据保护是 Amazon 的一项主要新功能 SNS
- MDP用于扫描邮件中是否有机密或敏感信息
- 对流经主题的所有内容提供消息审计
- 对发布到主题的消息和主题传输的消息提供内容访问控制

> **Important**
>
> 我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括您使用控制台、、或使用亚马逊SNS或其他 Amazon Web Ser API v AWS CLI ices 时 AWS SDKs。您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。如果您URL向外部服务器提供，我们强烈建议您不要在中包含凭据信息，URL以验证您对该服务器的请求。

以下各节提供了有关Amazon数据保护的更多信息SNS。

主题

- 亚马逊SNS数据加密
- 使用VPC终端节点保护 Amazon SNS 流量
- 通过消息数据保护增强 Amazon 的SNS安全

亚马逊SNS数据加密

数据保护是指保护传输中（往返亚马逊时SNS）和静态数据（存储在亚马逊SNS数据中心的磁盘上时）的数据。您可以使用安全套接字层 (SSL) 或客户端加密来保护传输中的数据。默认情况下，Amazon 使用磁盘加密SNS存储消息和文件。您可以通过请求 Amazon 在将消息保存SNS到其数据中心的加密文件系统之前对其进行加密来保护静态数据。亚马逊SNS建议使用SSE来优化数据加密。

主题

- 使用服务器端加密保护 Amazon SNS 数据
- 管理 Amazon SNS 加密密钥和成本
- 使用服务器端加密设置 Amazon SNS 主题加密
- 使用加密的亚马逊SQS队列订阅设置亚马逊SNS主题加密

使用服务器端加密保护 Amazon SNS 数据

服务器端加密 (SSE) 允许您使用在 AWS Key Management Service (AWS KMS) 中管理的密钥保护 Amazon 主题中的消息内容，从而将敏感数据存储在加密SNS主题中。

SSEAmazon SNS 收到消息后立即对其进行加密。消息以加密形式存储，仅在发送时才解密。

- 有关使用 AWS Management Console 或 AWS SDK for Java （通过SSE使用`CreateTopic`和`SetTopicAttributes`API操作设置`KmsMasterKeyId`属性）进行管理的信息，请参见使用服务器端加密设置 Amazon SNS 主题加密。
- 有关使用 AWS CloudFormation （通过使用`AWS::SNS::Topic`资源设置`KmsMasterKeyId`属性）创建加密主题的信息，请参阅《AWS CloudFormation 用户指南》。

> **Important**
>
> 对SSE启用的主题的所有请求都必须使用HTTPS签名版本 4。
> 有关其他服务与加密主题的兼容性的信息，请参阅服务文档。
> Amazon SNS 仅支持对称加密KMS密钥。您不能使用任何其他类型的KMS密钥来加密您的服务资源。有关确定密钥是否为对称加密KMS密钥的帮助，请参阅识别非对称KMS密钥。

AWS KMS 将安全、高度可用的硬件和软件相结合，提供可扩展到云端的密钥管理系统。当您将 Amazon SNS 与一起 AWS KMS使用时，加密您的消息数据的数据密钥也会被加密并与其保护的数据一起存储。

使用 AWS KMS具有以下好处：

- 您可以自行创建和管理 AWS KMS key。
- 您也可以为 Amazon 使用 AWS托管KMS密钥SNS，这些密钥对于每个账户和地区都是唯一的。
- AWS KMS 安全标准可以帮助您满足与加密相关的合规性要求。

有关更多信息，请参阅什么是 AWS Key Management Service？ 在《AWS Key Management Service 开发人员指南》中。

主题

- 加密范围
- 关键术语

加密范围

SSE对 Amazon SNS 主题中的消息正文进行加密。

SSE 不对以下各项进行加密：

- 主题元数据（主题名称和属性）
- 消息元数据（主题、消息 ID、时间戳和属性）
- 数据保护策略
- 每个主题的指标数

> **Note**
>
> - 仅在启用主题加密后发送消息时对其进行加密。Amazon SNS 不对积压的消息进行加密。
> - 任何加密的消息将保持加密状态，即使已禁用其主题的加密。

关键术语

以下关键术语可以帮助您更好地理解的功能SSE。有关详细说明，请参阅《Amazon 简单通知服务API参考》。

数据密钥

负责加密 Amazon SNS 消息内容的数据加密密钥 (DEK)。

有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的数据密钥和 AWS Encryption SDK 开发人员指南中的信封加密。

AWS KMS key ID

您的账户或其他账户 AWS KMS中的别名ARN、别名 AWS KMS key、密钥 ARN ID 或密钥，或自定义密钥。虽然 Amazon AWS 托管 AWS KMS 的别名始终SNS`alias/aws/sns`是，但自定义的别名 AWS KMS 可以是`alias/MyAlias`。您可以使用这些 AWS KMS 密钥来保护 Amazon SNS 主题中的消息。

> **Note**
>
> 记住以下内容：

- 首次使用KMS为主题指定亚马逊 AWS 托管时， AWS KMS 会创建SNSKMS适用于亚马逊的 AWS 托管SNS。 AWS Management Console
- 或者，在您首次对SSE启用状态的主题使用Publish操作时， AWS KMS 会创建KMS适用于 Amazon 的 AWS 托管SNS。

您可以创建 AWS KMS 密钥，定义控制 AWS KMS 密钥使用方式的策略，并 AWS KMS 使用 AWS KMS 控制台的AWS KMS keys部分或CreateKey AWS KMS 操作来审计使用情况。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS keys和创建密钥。有关 AWS KMS 标识符的更多示例，请参阅AWS Key Management Service API参考文献KeyId中的。有关查找 AWS KMS 标识符的信息，请参阅查找密钥 ID 和ARN《AWS Key Management Service 开发者指南》。

Important

使用需要支付额外费用 AWS KMS。有关更多信息，请参阅 估算成本 AWS KMS 和 AWS Key Management Service 定价。

管理 Amazon SNS 加密密钥和成本

以下部分提供了有关使用 AWS Key Management Service (AWS KMS) 中托管的密钥的信息。了解更多信息

Note

Amazon SNS 仅支持对称加密KMS密钥。您不能使用任何其他类型的KMS密钥来加密您的服务资源。有关确定密钥是否为对称加密KMS密钥的帮助，请参阅识别非对称KMS密钥。

主题

- 估算成本 AWS KMS
- 配置 AWS KMS 权限
- AWS KMS 错误

估算成本 AWS KMS

为了预测成本并更好地了解您的 AWS 账单，您可能需要了解Amazon SNS 使用您的账单的频率 AWS KMS key。

> **Note**
>
> 尽管以下公式可以让你很好地了解预期成本，但由于 Amazon 的分布式特性，实际成本可能会更高SNS。

要计算每个主题的API请求数 (R)，请使用以下公式：

```
R = B / D * (2 * P)
```

B 是账单周期（以秒为单位）。

D是数据密钥的重复使用周期（以秒为单位，Amazon 最多可SNS重复使用数据密钥 5 分钟）。

P是发送到 Amazon SNS 主题的出版主体人数。

以下是一些示例计算。有关准确的定价信息，请参阅 AWS Key Management Service 定价。

示例 1：计算 1 个发布者和 1 个主题的 AWS KMS API呼叫次数

此示例假定：

- 账单周期为 1 月 1 日 - 31 日（2678400 秒）。
- 数据密钥重用周期为 5 分钟（300 秒）。
- 提供了 1 个主题。
- 提供了 1 个发布委托人。

```
2,678,400 / 300 * (2 * 1) = 17,856
```

示例 2：计算多个发布者和 2 个主题的 AWS KMS API调用次数

此示例假定：

- 账单周期为 2 月 1 日 - 28 日（2419200 秒）。

- 数据密钥重用周期为 5 分钟（300 秒）。
- 提供了 2 个主题。
- 第一个主题具有 3 个发布委托人。
- 第二个主题具有 5 个发布委托人。

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

配置 AWS KMS 权限

在使用之前SSE，必须将 AWS KMS key 策略配置为允许对主题进行加密以及对消息进行加密和解密。有关 AWS KMS 权限的示例和更多信息，请参阅《AWS Key Management Service 开发者指南》中的 “AWS KMS API权限：操作和资源参考”。有关如何使用服务器端加密设置 Amazon SNS 主题的详细信息，请参阅设置使用服务器端加密的 Amazon SNS 主题。

> **Note**
>
> 您还可以使用IAM策略管理对称加密KMS密钥的权限。有关更多信息，请参阅将IAM策略与一起使用 AWS KMS。
> 虽然您可以配置向亚马逊发送和接收的全局权限SNS，但 AWS KMS 需要KMSs在IAM策略`Resource`部分明确命名特定区域的完整ARN内容。

您还必须确保的密钥策略 AWS KMS key 允许必要的权限。为此，请在KMS密钥策略中将在 Amazon 中生成和使用加密消息的委托人命名SNS为用户。

或者，您可以在分配给在 Amazon KMS ARN SNS 中发布和订阅以接收加密消息的委托人的IAM策略中指定所需的 AWS KMS 操作。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的管理对 AWS KMS的访问。

如果您为 Amazon SNS 主题选择了客户管理的密钥，并且您使用别名使用IAM策略或带有条件KMS密钥的密KMS钥策略来控制对密钥的访问`kms:ResourceAliases`，请确保所选的客户管理的密钥也具有关联的别名。有关使用别名控制KMS密钥访问权限的更多信息，请参阅《AWS Key Management Service 开发者指南》中的使用别名控制KMS密钥访问权限。

允许用户使用以下方式向主题发送消息 SSE

发布者必须具有 AWS KMS key的 `kms:GenerateDataKey*` 和 `kms:Decrypt` 权限。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-
east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }, {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

启用来自 AWS 服务的事件源和加密主题之间的兼容性

有几项 AWS 服务会向 Amazon SNS 主题发布事件。要允许这些事件源使用加密主题，您必须执行以下步骤。

1. 使用客户自主管理型密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。
2. 要允许 AWS 服务拥有`kms:GenerateDataKey*`和`kms:Decrypt`权限，请在KMS策略中添加以下语句。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

事件源	服务主体
Amazon CloudWatch	cloudwatch.amazonaws.com
亚马逊 CloudWatch 活动	events.amazonaws.com
AWS CodeCommit	codecommit.amazonaws.com
AWS CodeStar	codestar-notifications.amazonaws.com
AWS Database Migration Service	dms.amazonaws.com
AWS Directory Service	ds.amazonaws.com
Amazon DynamoDB	dynamodb.amazonaws.com
Amazon Inspector	inspector.amazonaws.com
Amazon Redshift	redshift.amazonaws.com
Amazon RDS	events.rds.amazonaws.com
Amazon S3 Glacier	glacier.amazonaws.com
Amazon Simple Email Service	ses.amazonaws.com
Amazon Simple Storage Service	s3.amazonaws.com
AWS Snowball	importexport.amazonaws.com
AWS Systems Manager 事件管理器	AWS Systems Manager 事件管理器包含两个服务原则：ssm-incidents.amazonaws.com ；ssm-contacts.amazonaws.com

> **Note**
>
> 某些 Amazon SNS 事件源要求您在 AWS KMS key 策略中提供IAM角色（而不是服务主体）：
>
> - Amazon A EC2 uto Scaling
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2 映像生成器

3. 在KMS资源策略中添加`aws:SourceAccount`和`aws:SourceArn`条件密钥，以进一步保护密KMS钥免受混淆的副手攻击。有关每种案例的具体详细信息，请参阅上述特定于服务的文档列表。

> **Important**
>
> EventBridge加密主题不支持在 AWS KMS 策略中添加`aws:SourceAccount`和`aws:SourceArn`。

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "service.amazonaws.com"
  },
  "Action": [
    "kms:GenerateDataKey*",
    "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "customer-account-id"
    },
```

```
    "ArnLike": {
      "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type:customer-resource-id"
    }
  }
}
```

4. 使用@@ 您的主题启SSE用KMS。
5. 向ARN事件源提供加密主题的。

AWS KMS 错误

当您与 Amazon SNS 合作时 AWS KMS，可能会遇到错误。以下列表描述了这些错误和可能的故障排除解决方案。

KMSAccessDeniedException

密文引用了不存在的或您无权访问的密钥。

HTTP状态码：400

KMSDisabledException

请求被拒绝，因为指定的KMS未启用。

HTTP状态码：400

KMSInvalidStateException

由于指定资源的状态对此请求无效，请求被拒绝。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS keys的密钥状态。

HTTP状态码：400

KMSNotFoundException

由于找不到指定的实体或资源，请求被拒绝。

HTTP状态码：400

KMSOptInRequired

AWS 访问密钥 ID 需要订阅该服务。

HTTP状态码：403

KMSThrottlingException

由于请求限制而导致请求被拒绝。有关节流的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的限额。

HTTP状态码：400

使用服务器端加密设置 Amazon SNS 主题加密

使用服务器端加密 (SSE)，您可以将敏感数据存储在加密的主题中。SSE使用 AWS Key Management Service (AWS KMS) 中管理的密钥保护 Amazon SNS 主题中的消息内容。有关使用 Amazon 进行服务器端加密的更多信息SNS，请参阅使用服务器端加密保护 Amazon SNS 数据。有关创建 AWS KMS 密钥的更多信息，请参阅《AWS Key Management Service 开发者指南》中的创建密钥。

> **Important**
>
> 对SSE启用的主题的所有请求都必须使用HTTPS签名版本 4。

使用 Amazon SNS 主题启用服务器端加密 (SSE) AWS Management Console

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 在主题页面上，选择一个主题，然后选择操作和编辑。
4. 展开加密部分并执行以下操作：

   a. 选择 Enable encryption（启用加密）。

   b. 指定密 AWS KMS 钥。有关更多信息，请参阅 关键术语。

   对于每KMS种类型，都会显示描述、账户KMSARN和。

   > **Important**
   >
   > 如果您不是的所有者KMS，或者您使用不具有`kms:ListAliases`和`kms:DescribeKey`权限的账户登录，则将无法在 Amazon SNS 控制台KMS上查看相关信息。
   > 要求的所有者授KMS予您这些权限。有关更多信息，请参阅《AWS Key Management Service 开发者指南》中的 “AWS KMS API权限：操作和资源参考”。

- 默认选择亚马逊 AWS 托管 KMSSNS（默认）别名/aws/sns。

  > **Note**
  >
  > 记住以下内容：
  >
  > - 首次使用KMS为主题指定亚马逊 AWS 托管时， AWS KMS 会创建SNSKMS适用于亚马逊的 AWS 托管SNS。 AWS Management Console
  > - 或者，当您首次对SSE启用主题使用Publish操作时， AWS KMS 会创建KMS适用于 Amazon 的 AWS 托管SNS。

- 要使用 AWS 账户KMS中的自定义项，请选择KMS关键字段，然后KMS从列表中选择自定义字段。

  > **Note**
  >
  > 有关创建自定义密钥的说明KMSs，请参阅《AWS Key Management Service 开发者指南》中的创建密钥

- 要使用您 AWS 账户或其他账户KMSARN中的自定义设置，请在KMS密钥字段中输入该自定义项。 AWS

5. 选择 Save changes（保存更改）。

   SSE已为你的主题启用而且 ***MyTopic***页面显示出来。

   主题的加密状态、 AWS 账户、客户主密钥 (CMK) 和描述显示在加密选项卡上。CMKARN

设置使用服务器端加密的 Amazon SNS 主题

创建KMS密钥时，请使用以下KMS密钥策略：

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "service.amazonaws.com"
     },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
```

```
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type/customer-resource-id"
        },
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:customer-account-id:your_sns_topic_name"
        }
    }
}
```

对消费者的影响

为某个 Amazon SNS 主题启用后，订阅者的消息消费过程将保持不变。SSE AWS 使用管理加密和解密过程。 AWS KMS因此，订阅者无需对其现有设置进行任何更改即可处理加密消息。 AWS 确保消息在静态状态下进行加密，并在发送给订阅者之前自动解密。这意味着订阅者将继续像启用加密之前一样接收和处理消息，无需任何额外的配置或解密逻辑。此外， AWS 建议使用HTTPS来确保消息的安全传输。

使用加密的亚马逊SQS队列订阅设置亚马逊SNS主题加密

您可以为主题启用服务器端加密 (SSE) 以保护其数据。要允许亚马逊SNS向加密的亚马逊SQS队列发送消息，与亚马逊队SQS列关联的客户托管密钥必须有政策声明，授予亚马逊SNS服务主体访问 AWS KMS API操作`GenerateDataKey`和`Decrypt`。有关使用的更多信息SSE，请参阅使用服务器端加密保护 Amazon SNS 数据。

本页介绍如何使用SSE为订阅了加密亚马逊SQS队列的亚马逊SNS主题启用该 AWS Management Console主题。

步骤 1：创建自定义KMS密钥

1. 通过至少具有 `AWSKeyManagementServicePowerUser` 策略的用户登录 AWS KMS 控制台。
2. 选择 Create a key (创建密钥)。
3. 要创建对称加密KMS密钥，请在 “密钥类型” 中选择 “对称”。

   有关如何在 AWS KMS 控制台中创建非对称KMS密钥的信息，请参阅创建非对称KMS密钥（控制台）。
4. 在 Key usage（密钥用法）中，已为您选择了 Encrypt and decrypt（加密和解密）选项。

有关如何创建生成和验证MAC代码的KMS密钥的信息，请参阅创建HMACKMS密钥。

有关高级选项的更多信息，请参阅专用密钥。

5. 选择下一步。
6. 键入KMS密钥的别名。别名名称不能以 **aws/** 开头。该**aws/**前缀由 Amazon Web Services 保留，用于 AWS 托管式密钥 在您的账户中表示。

> **Note**
>
> 添加、删除或更新别名可以允许或拒绝对KMS密钥的权限。有关详细信息，ABAC请参阅了解 AWS KMS和使用别名控制对KMS密钥的访问权限。

别名是可用于标识KMS密钥的显示名称。我们建议您选择一个别名，以表明您计划保护的数据类型或计划与KMS密钥一起使用的应用程序。

在中创建KMS密钥时需要使用别名。 AWS Management Console当您使用CreateKey操作时，它们是可选的。

7. （可选）键入KMS密钥的描述。

现在，除非密钥状态为 `Pending Deletion` 或 `Pending Replica Deletion`，否则您可以随时添加描述或更新描述。要添加、更改或删除现有客户托管密钥的描述，请在中编辑描述 AWS Management Console 或使用UpdateKeyDescription操作。

8. (可选) 键入标签键和一个可选标签值。要向KMS密钥添加多个标签，请选择添加标签。

> **Note**
>
> 标记或取消标记KMS密钥可以允许或拒绝对密钥的权限。KMS有关详细信息，ABAC请参阅了解 AWS KMS和使用标签控制对KMS密钥的访问权限。

向 AWS 资源添加标签时， AWS 会生成一份成本分配报告，其中包含按标签汇总的使用量和成本。标签还可用于控制对KMS密钥的访问。有关为KMS密钥添加标签的信息，请参阅为密钥添加标签和。ABAC AWS KMS

9. 选择下一步。
10. 选择可以管理KMS密钥的IAM用户和角色。

> **Note**
>
> 此密钥策略赋予了对该KMS密钥的 AWS 账户 完全控制权。它允许账户管理员使用IAM策略向其他委托人授予管理KMS密钥的权限。有关详细信息，请参阅原定设置密钥策略。
>
> IAM最佳做法不鼓励使用具有长期凭证的IAM用户。只要有可能，就使用提供临时证书的IAM角色。有关详细信息，请参阅《IAM用户指南》IAM中的安全最佳实践。

11. （可选）要防止所选IAM用户和角色删除此KMS密钥，请在页面底部的密钥删除部分，清除“允许密钥管理员删除此密钥”复选框。
12. 选择下一步。
13. 选择可以在加密操作中使用密钥的IAM用户和角色。选择下一步。
14. 在 Review and edit key policy (审核和编辑密钥策略) 页面上，向密钥策略添加以下语句，然后选择 Finish (完成)。

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

新的客户自主管理型密钥将显示在密钥列表中。

第 2 步：创建加密的 Amazon SNS 主题

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 选择创建主题。
4. 在 Create new topic (创建新主题) 页面上，对于 Name (名称)，输入主题名称（例如，`MyEncryptedTopic`），然后选择 Create topic (创建主题)。

5. 展开加密部分并执行以下操作：

   a. 选择 Enable server-side encryption (启用服务器端加密)。

   b. 指定客户自主管理型密钥。有关更多信息，请参阅 关键术语。

      对于每种客户托管密钥类型，都会显示描述、账户和客户托管密钥ARN。

      > **Important**
      >
      > 如果您不是客户托管密钥的所有者，或者您使用不具有`kms:ListAliases`和`kms:DescribeKey`权限的账户登录，则将无法在 Amazon SNS 控制台上查看有关客户托管密钥的信息。
      > 要求客户自主管理型密钥的拥有者授予您这些权限。有关更多信息，请参阅《AWS Key Management Service 开发者指南》中的 “AWS KMS API权限：操作和资源参考”。

   c. 对于客户托管密钥 MyCustomKey，请选择您之前创建的密钥，然后选择启用服务器端加密。

6. 选择 Save changes（保存更改）。

   SSE已为你的主题启用并显示MyTopic页面。

   主题的加密状态、 AWS 账户、客户管理的密钥、客户管理的密钥ARN和描述显示在加密选项卡上。

您的新的加密主题将显示在主题列表中。

第 3 步：创建和订阅加密的 Amazon SQS 队列

1. 登录 Amazon SQS 控制台。
2. 选择 Create New Queue（创建队列）。
3. 在 Create New Queue (创建新队列) 页面上，执行以下操作：

   a. 输入 Queue Name (队列名称)（例如，`MyEncryptedQueue1`）。

   b. 选择 Standard Queue (标准队列)，然后选择 Configure Queue (配置队列)。

   c. 选择 “使用” SSE。

   d. 对于 AWS KMS keyMyCustomKey，选择您之前创建的队列，然后选择创建队列。

4. 重复该过程以创建第二个队列（例如，名为 `MyEncryptedQueue2`）。

您的新的加密队列将显示在队列列表中。

5. 在 Amazon SQS 控制台上MyEncryptedQueue1，选择队列操作，MyEncryptedQueue2然后选择队列订阅SNS主题。
6. 在 “订阅主题” 对话框中，选择 “选择主题” MyEncryptedTopic，然后选择 “订阅”。

   您的加密队列对加密主题的订阅将显示在 Topic Subscription Result (主题订阅结果) 对话框中。
7. 选择确定。

步骤 4：向加密的主题发布消息

1. 登录 Amazon SNS 控制台。
2. 在导航面板上，选择 Topics（主题）。
3. 从主题列表中选择，MyEncryptedTopic然后选择发布消息。
4. 在 Publish a message (发布消息) 页面上，执行以下操作：
   a. （可选）在 Message details (消息详细信息) 部分中，输入 Subject (主题)（例如：`Testing message publishing`）。
   b. 在 Message body (消息正文) 部分中，输入消息正文（例如，`My message body is encrypted at rest.`）。
   c. 选择发布消息。

您的消息将发布到订阅的加密队列。

步骤 5：验证消息传输

1. 登录 Amazon SQS 控制台。
2. 从队列列表中选择 MyEncryptedQueue1，然后选择发送和接收消息。
3. 在 “在 MyEncryptedQueue1 页中发送和接收消息” 上，选择 “轮询留言”。

   此时将显示您之前发送的消息。
4. 选择 More Details (更多详细信息) 以查看您的消息。
5. 完成后，选择 Close (关闭)。
6. 对 MyEncryptedQueue2 重复该过程。

使用VPC终端节点保护 Amazon SNS 流量

亚马逊虚拟私有云 (AmazonVPC) 终端节点SNS是其中的一个逻辑实体VPC，它只允许连接到亚马逊SNS。它们会将请求VPC路由到 Amazon SNS 并将响应路由回VPC。以下各节提供有关使用VPC终端节点和创建终VPC端节点策略的信息。

如果您使用亚马逊虚拟私有云 (AmazonVPC) 托管您的 AWS 资源，则可以在您VPC和亚马逊之间建立私有连接SNS。通过此连接，您可以将消息发布到您的 Amazon SNS 主题，而无需通过公共互联网发送。

Amazon VPC 是一项 AWS 服务，可用于在您定义的虚拟网络中启动 AWS 资源。使用 aVPC，您可以控制自己的网络设置，例如 IP 地址范围、子网、路由表和网络网关。要将您连接VPC到AmazonSNS，您需要定义一个接口VPC终端节点。这种类型的终端节点使您能够VPC连接到 AWS 服务。该终端节点SNS无需互联网网关、网络地址转换 (NAT) 实例或连接，即可提供与 Amazon 的可靠、可扩展的VPN连接。有关更多信息，请参阅 Amazon VPC 用户指南中的接口VPC终端节点。

本节中的信息仅供Amazon用户使用VPC。要了解更多信息并开始创建VPC，请参阅《亚马逊VPC用户指南》VPC中的 “亚马逊入门”。

> **Note**
>
> VPC终端节点不允许您将亚马逊SNS主题订阅到私有 IP 地址。

主题

- 为亚马逊创建亚马逊VPC终端节点 SNS
- 为亚马逊创建亚马逊VPC终端节点策略 SNS
- 发布来自亚马逊的亚马逊SNS消息 VPC

为亚马逊创建亚马逊VPC终端节点 SNS

要从亚马逊向您的亚马逊SNS主题发布消息VPC，请创建一个接口VPC终端节点。然后，您可以向您的主题发布消息，同时将流量保留在您管理的网络中VPC。

使用以下信息创建终端节点并测试您与 Amazon VPC 之间的连接SNS。或者，有关可帮助您从头开始的演练，请参阅发布来自亚马逊的亚马逊SNS消息 VPC。

创建终端节点

您可以VPC使用、、、a、Amazon 或 AWS Management Console AWS CLI，在您的中创建亚马逊SNSSNSAPI终端节点 AWS CloudFormation。 AWS SDK

有关使用亚马逊VPC控制台或创建和配置终端节点的信息 AWS CLI，请参阅亚马逊VPC用户指南中的创建接口终端节点。

> **Important**
>
> 您只能在亚马逊SNS终端节点上使用亚马逊 Virtual HTTPS Private Cloud。
> 创建终端节点时，请指定 Amazon SNS 作为您VPC想要连接的服务。在 Amazon VPC 控制台中，服务名称因地区而异。例如，如果您选择美国东部（弗吉尼亚北部），则服务名称为 com.amazonaws.us-east-1.sns。
> 当您将 Amazon 配置SNS为从亚马逊发送消息时VPC，必须启用私有功能DNS并按照格式指定终端节点`sns.us-east-2.amazonaws.com`。
> Pri DNS vate 不支持旧版端点，例如`queue.amazonaws.com`或`us-east-2.queue.amazonaws.com`。

有关使用创建和配置端点的信息 AWS CloudFormation，请参阅《AWS CloudFormation 用户指南》中的`AWS::EC2::VPCEndpoint`资源。

测试您与 Amazon VPC 之间的连接 SNS

为亚马逊创建终端节点后SNS，您可以将您的消息发布VPC到您的亚马逊SNS主题。要测试此连接，请执行以下操作：

1. Connect 连接到驻留在您的中的 Amazon EC2 实例VPC。有关连接的信息，请参阅亚马逊EC2文档中的连接到您的 Linux 实例或连接到您的 Windows 实例。

   例如，要使用SSH客户端连接到 Linux 实例，请从终端运行以下命令：

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   其中：

   - ec2-key-pair.pem 是包含您在创建实例时亚马逊EC2提供的密钥对的文件。
   - instance-hostname 是实例的公有主机名。要在 Amazon EC2 控制台中获取主机名，请执行以下操作：选择实例，选择您的实例，然后找到 P ublic DNS (IPv4) 的值。

2. 在您的实例中，使用 Amazon SNS publish命令和 AWS CLI。您可以使用以下命令将简单的消息发送到主题：

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

其中：

- aw s-region AWS 是主题所在的区域。
- sns-topic-arn是该主题的亚马逊资源名称 (ARN)。要ARN从 Amazon SNS 控制台获取，请执行以下操作：选择主题，找到您的主题，然后在ARN列中找到值。

如果 Amazon 成功接收消息SNS，则终端会打印一个消息 ID，如下所示：

```
{
   "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

为亚马逊创建亚马逊VPC终端节点策略 SNS

您可以为亚马逊VPC终端节点创建策略，SNS在其中指定以下内容：

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅 Amazon VPC 用户指南中的使用VPC终端节点控制对服务的访问。

以下示例VPC终端节点策略指定允许IAM用户MyUser向 Amazon SNS 主题发布内容MyTopic。

```
{
  "Statement": [{
    "Action": ["sns:Publish"],
    "Effect": "Allow",
    "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
    "Principal": {
      "AWS": "arn:aws:iam:123456789012:user/MyUser"
    }
  }]
```

```
}
```

以下各项将被拒绝：

- 其他 Amazon SNS API 操作，例如`sns:Subscribe`和`sns:Unsubscribe`。
- 尝试使用此VPC端点的其他IAM用户和规则。
- `MyUser`发布到另一个 Amazon SNS 主题。

> **Note**
>
> IAM用户仍然可以在外部使用其他 Amazon SNS API 操作VPC。

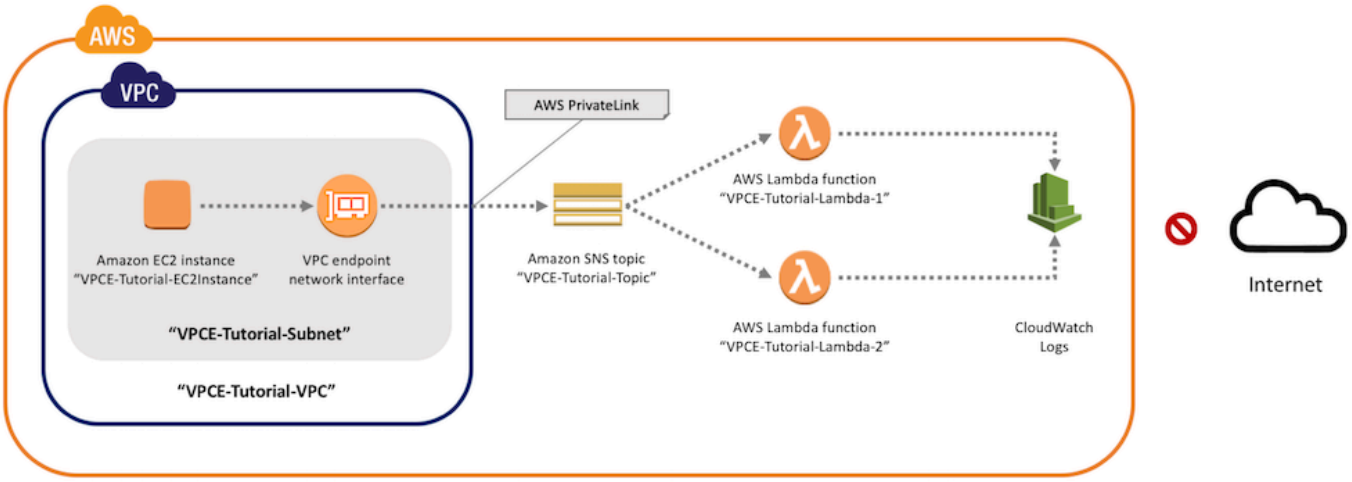
发布来自亚马逊的亚马逊SNS消息 VPC

本节介绍如何在 Amazon SNS 主题上发布消息，同时确保私有网络中的消息安全。您发布来自托管在亚马逊虚拟私有云 (AmazonVPC) 中的亚马逊EC2实例的消息。消息保留在 AWS 网络中，无需通过公共互联网传输。通过私密发布来自 a 的消息VPC，您可以提高应用程序与 Amazon 之间流量的安全性SNS。当您发布有关客户的个人身份信息 (PII) 或您的应用程序受市场法规约束时，这种安全性非常重要。例如，如果您的医疗系统必须符合《健康保险便携性和问责法》(HIPAA)，或者金融系统必须符合支付卡行业数据安全标准 (PCIDSS)，则私下发布会很有帮助。

常见步骤如下：

- 使用 AWS CloudFormation 模板在您的中自动创建临时专用网络 AWS 账户。
- 创建将VPC与 Amazon 连接起来的VPC终端节点SNS。
- 登录亚马逊EC2实例，私下向亚马逊SNS主题发布消息。
- 验证消息是否已成功发送。
- 删除您在此过程中创建的资源，这样它们就不会保留在您的资源中 AWS 账户。

下图描述了您在完成这些步骤时在 AWS 账户中创建的专用网络：



该网络由包含 Amazon EC2 实例的网络组成。VPC该实例SNS通过接口VPC终端节点连接到Amazon。这种类型的端点连接到由提供支持的服务 AWS PrivateLink。建立此连接后，即使网络已与公共互联网断开连接，您也可以登录亚马逊EC2实例并向亚马逊SNS主题发布消息。该主题将其收到的消息分散到两个订阅 AWS Lambda 函数。这些函数将它们收到的消息记录在 Amazon CloudWatch 日志中。

完成这些步骤大约需要 20 分钟。

主题

开始前的准备工作

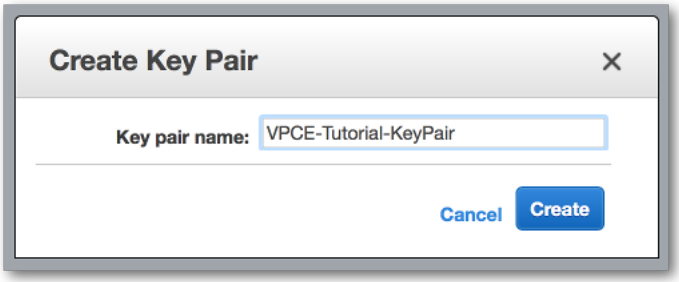
在开始之前，您需要 Amazon Web Services (AWS) 账户。当您注册时，您的账户会自动注册所有服务AWS，包括亚马逊SNS和亚马逊VPC。如果您尚未创建账户，请前往 https://aws.amazon.com/，然后选择 Create a Free Account（创建免费账户）。

第 1 步：创建亚马逊EC2密钥对

密钥对用于登录亚马逊EC2实例。它包含一个用于加密您的登录信息的公有密钥和一个用于解密该信息的私有密钥。创建密钥对时，下载该私有密钥的副本。稍后，您可以使用 key pair 登录到 Amazon EC2 实例。要登录，请指定密钥对的名称，然后提供密钥对。

创建密钥对

1. 登录 AWS Management Console 并打开 Amazon EC2 控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在左侧导航菜单中，找到 Network & Security (网络与安全) 部分。然后，选择 Key Pairs (密钥对)。
3. 选择创建密钥对。
4. 在创建密钥对窗口中，在密钥对名称中，键入 **VPCE-Tutorial-KeyPair**。然后选择 Create。



5. 您的浏览器会自动下载私有密钥文件。将它保存至安全位置。Amazon EC2 为该文件提供的扩展名为 .pem。
6. （可选）如果您在 Mac 或 Linux 计算机上使用SSH客户端连接到您的实例，请使用chmod命令设置私钥文件的权限，以便只有您可以读取该文件：

   a. 打开终端并导航到包含该私有密钥的目录：

   ```
   $ cd /filepath_to_private_key/
   ```

   b. 使用以下命令设置权限：

   ```
   $ chmod 400 VPCE-Tutorial-KeyPair.pem
   ```

步骤 2：创建 AWS 资源

要设置基础架构，请使用 AWS CloudFormation 模板。模板是一种用作构建 AWS 资源（例如 Amazon EC2 实例和 Amazon SNS 主题）的蓝图的文件。此过程的模板已提供 GitHub 给您下载。

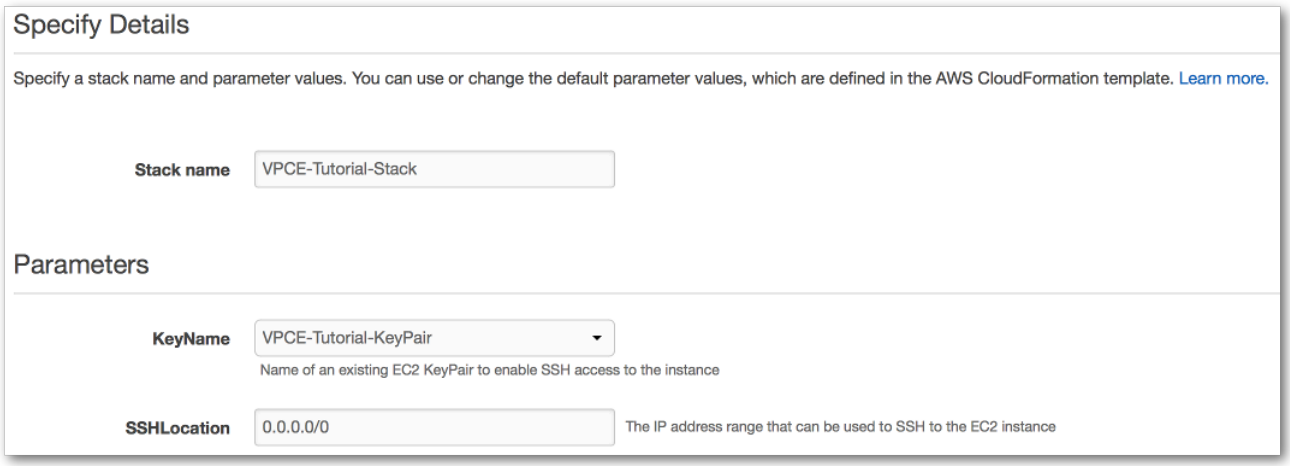
您可以向提供模板 AWS CloudFormation AWS CloudFormation ，并将所需的资源配置为堆栈 AWS 账户。堆栈是可作为单个单元管理的一系列资源。完成这些步骤后，您可以使用 AWS CloudFormation 一次性删除堆栈中的所有资源。除非您愿意 AWS 账户，否则这些资源不会保留在您的中。

此过程的堆栈包括以下资源：

- A VPC 和相关的网络资源，包括子网、安全组、互联网网关和路由表。
- 在子网中启动的 Amazon EC2 实例VPC。
- 亚马逊的SNS话题。
- 两个 AWS Lambda 功能。这些函数接收发布到 Amazon SNS 主题的消息，并在 CloudWatch 日志中记录事件。
- Amazon CloudWatch 指标和日志。
- 一个允许亚马逊EC2实例使用亚马逊的IAM角色SNS，以及一个允许 Lambda 函数写入日志的IAM角色。 CloudWatch

创建 AWS 资源

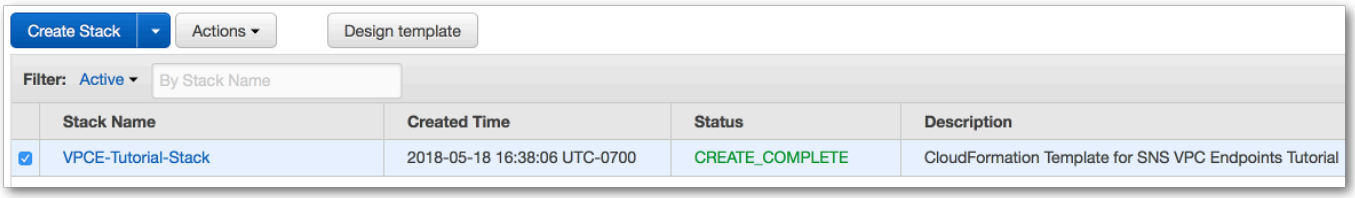
1. 从 GitHub 网站下载模板文件。
2. 登录 AWS CloudFormation 控制台。
3. 选择创建堆栈。
4. 在 Select Template (选择模板) 页面上，依次选择 Upload a template to Amazon S3 (将模板上传到 Amazon S3)、您的文件和下一步。
5. 在 Specify Details (指定详细信息) 页面上，指定堆栈和密钥名称：
   a. 对于 Stack name，键入 **`VPCE-Tutorial-Stack`**。
   b. 对于 KeyName，请选择 VPCE-教程-KeyPair。
   c. 对于 SSHLocation，保留默认值**`0.0.0.0/0`**。



d. 选择下一步。

6. 在 Options (选项) 页面上，保留所有默认值，然后选择 Next (下一步)。
7. 在 Review (审核) 页面上，验证堆栈详细信息。
8. 在 “权能” 下，确认 AWS CloudFormation 可能会使用自定义名称创建IAM资源。
9. 选择创建。

AWS CloudFormation 控制台打开堆栈页面。-Tutor VPCE ial-Stack 的状态为 _IN_。CREATE PROGRESS几分钟后，创建过程完成后，状态将变为 CREATE_ COMPLETE。



> **Tip**
>
> 选择刷新按钮以查看最新堆栈状态。

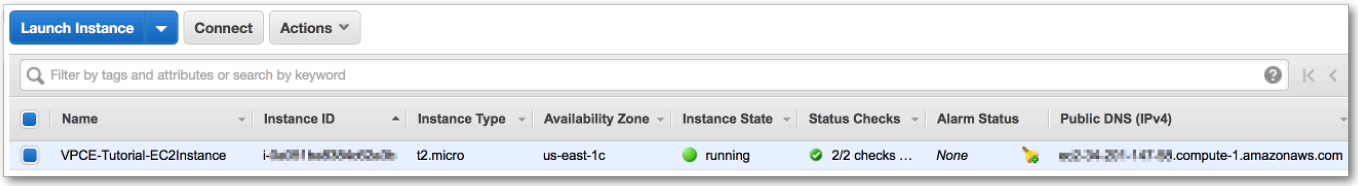
步骤 3：确认您的 Amazon EC2 实例无法访问互联网

在上一步中启动的 Amazon EC2 实例无法访问互联网。VPC它不允许出站流量，也无法向 Amazon SNS 发布消息。通过登录该实例验证此项。然后，尝试连接到公共终端节点，并尝试向 Amazon 发送消息SNS。

此时，发布操作尝试失败。在后面的步骤中，在您为 Amazon 创建VPC终端节点后SNS，您的发布尝试就会成功。

连接到您的 Amazon EC2 实例

1. 打开亚马逊EC2控制台，网址为https://console.aws.amazon.com/ec2/。
2. 在左侧导航菜单中，找到 Instances (实例) 部分。然后，选择 Instances (实例)。
3. 在实例列表中，选择 VPCE-Tutorial-EC2Instance。
4. 复制 P ublic DNS (IPv4) 列中提供的主机名。



5. 打开终端。在包含 key pair 的目录中，使用以下命令连接到实例，其中 *instance-hostname* 是您从 Amazon EC2 控制台复制的主机名：

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

验证实例是否缺少 Internet 连接

- 在您的终端中，尝试连接到任何公共终端节点，如 amazon.com：

```
$ ping amazon.com
```

由于连接尝试失败，您可以随时进行取消（Windows 上按 Ctrl + C 或 macOS 上按 Command + C）。

验证实例是否无法连接到 Amazon SNS

1. 登录 Amazon SNS 控制台。
2. 在左侧导航菜单中，选择 Topics (主题)。
3. 在主题页面上，复制主题 VPCE-教程主题的 Amazon 资源名称 (ARN)。
4. 在您的终端中，尝试向该主题发布消息：

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

由于发布尝试失败，您可以随时进行取消。

步骤 4：为亚马逊创建亚马逊VPC终端节点 SNS

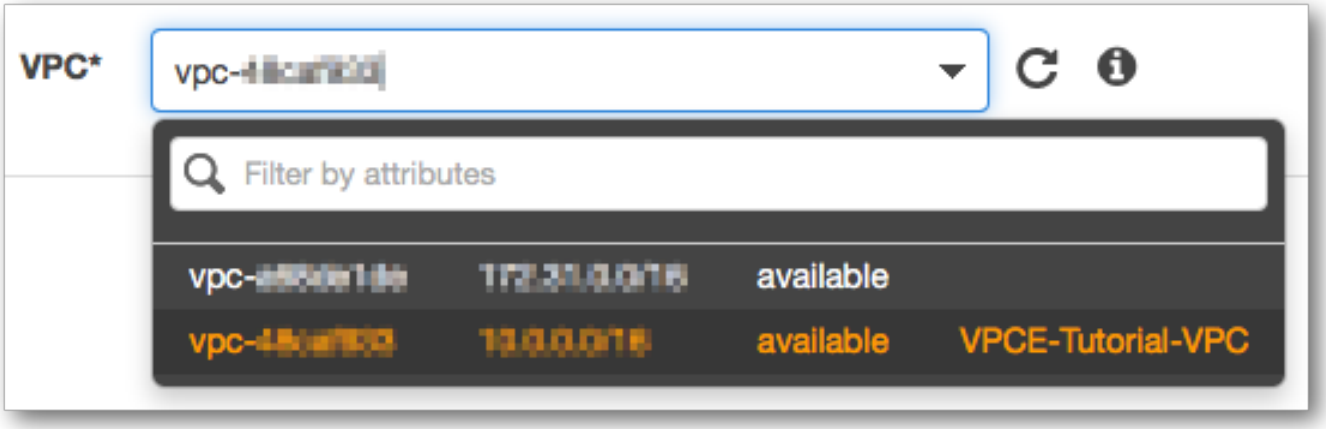
要将连接VPC到 AmazonSNS，您需要定义一个接口VPC终端节点。添加终端节点后，您可以登录自己的亚马逊EC2实例VPC，然后从那里使用亚马逊SNSAPI。您可以将消息发布到该主题，而且私下发布消息。它们停留在 AWS 网络中，并且不会在公共互联网上旅行。

> **Note**
>
> 该实例仍然无法访问互联网上的其他 AWS 服务和终端节点。

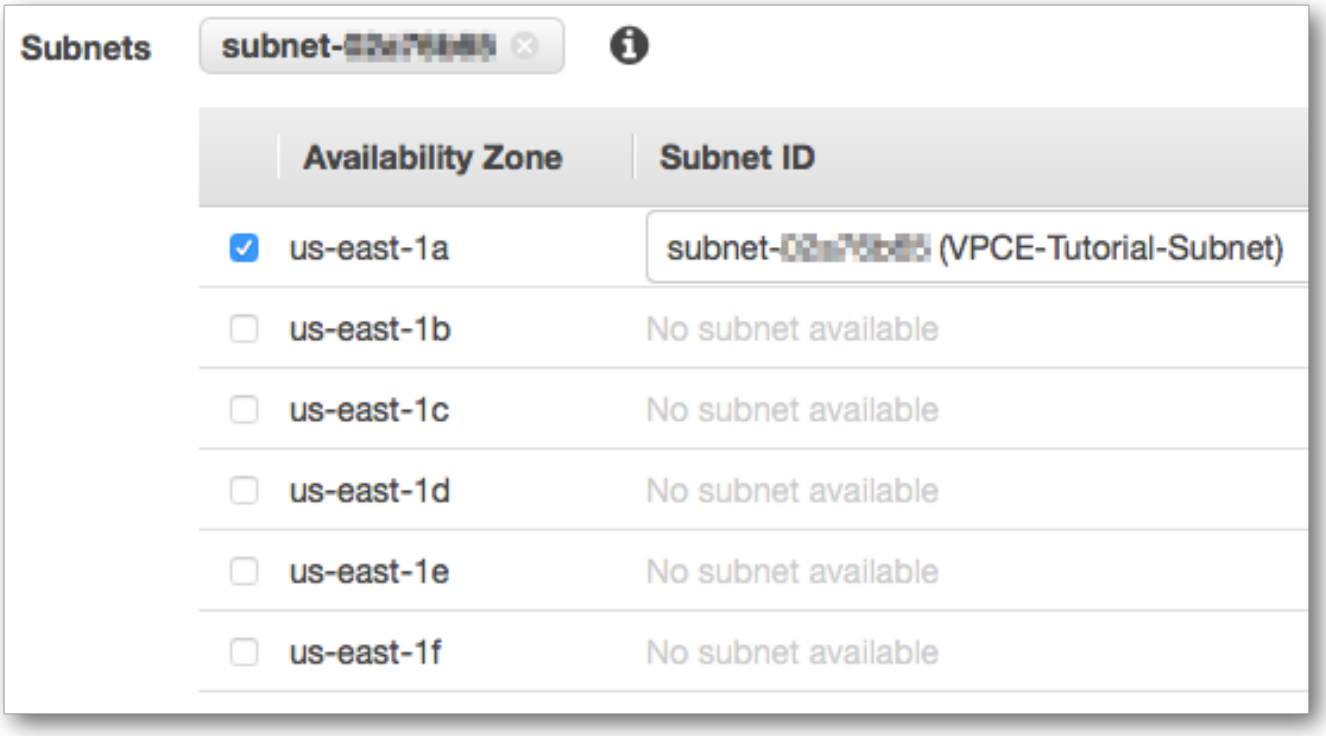
创建终端节点

1. 打开亚马逊VPC控制台，网址为https://console.aws.amazon.com/vpc/。
2. 在左侧导航菜单中，选择 Endpoints (终端节点)。
3. 选择 Create Endpoint（创建端点）。
4. 在 Create Endpoint（创建终端节点）页面上，对于 Service category（服务类别），保留默认选择 AWS 服务。
5. 在“服务名称”中，选择 Amazon 的服务名称SNS。

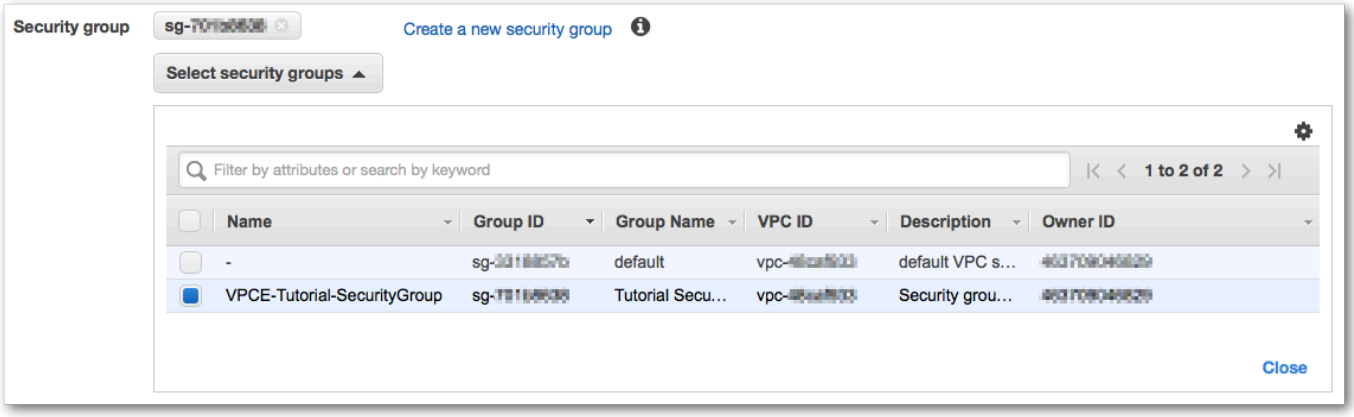
   服务名称因所选区域而异。例如，如果您选择美国东部（弗吉尼亚北部），则服务名称为 com.amazonaws。*us-east-1*.sns。
6. 对于 VPC，请选择名VPC为 VPCE-Tutorial-VPC 的。



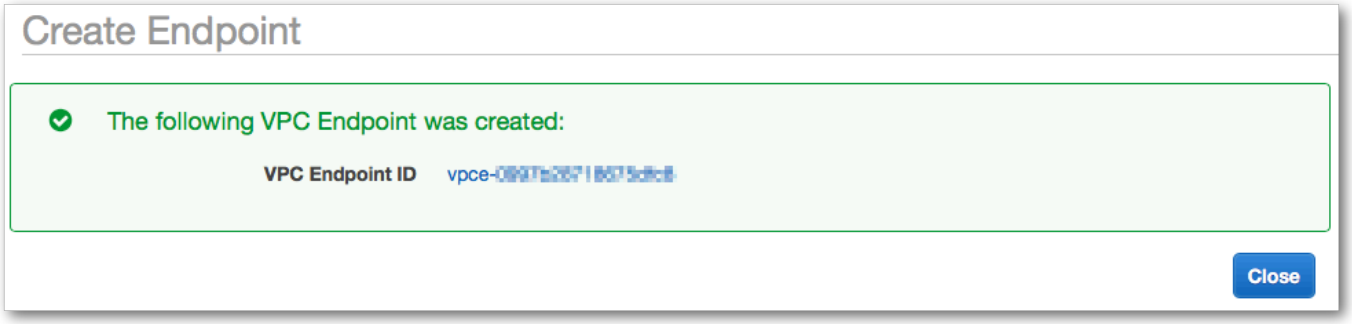
7. 对于子网，请选择子网 ID 中包含 VPCE-教程子网的子网。



8. 对于 “启用私有DNS名称”，选择 “为此端点启用”。
9. 对于安全组，选择选择安全组，然后选择 VPCE-教程-SecurityGroup。

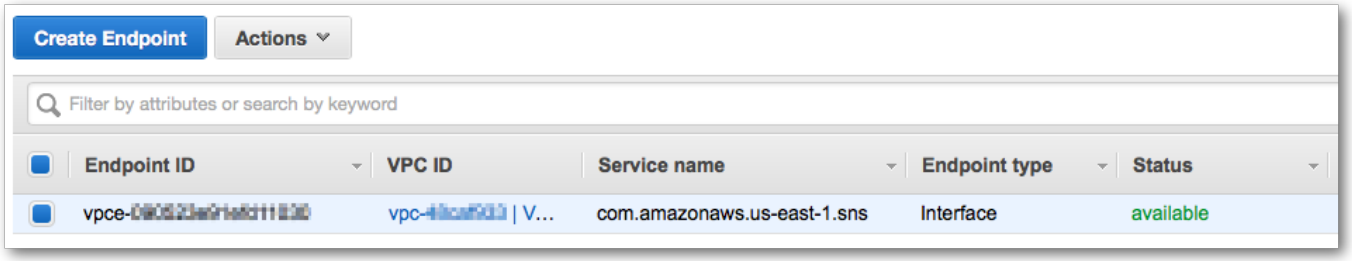


10. 选择创建端点。Amazon VPC 控制台确认VPC终端节点已创建。



11. 选择关闭。

    Amazon VPC 控制台打开终端节点页面。新终端节点的状态为 pending (待处理)。在创建过程完成后的几分钟内，该状态将变为 available (可用)。



第 5 步：向您的亚马逊SNS主题发布消息

现在，您已经VPC包含了 Amazon 的终端节点SNS，您可以登录亚马逊EC2实例并向该主题发布消息。

发布消息

1. 如果您的终端不再连接到您的 Amazon EC2 实例，请重新连接：

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. 运行与之前相同的命令，将消息发布到您的 Amazon SNS 主题。这次，发布尝试成功，Amazon SNS 返回了一个消息 ID：

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"

{
    "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

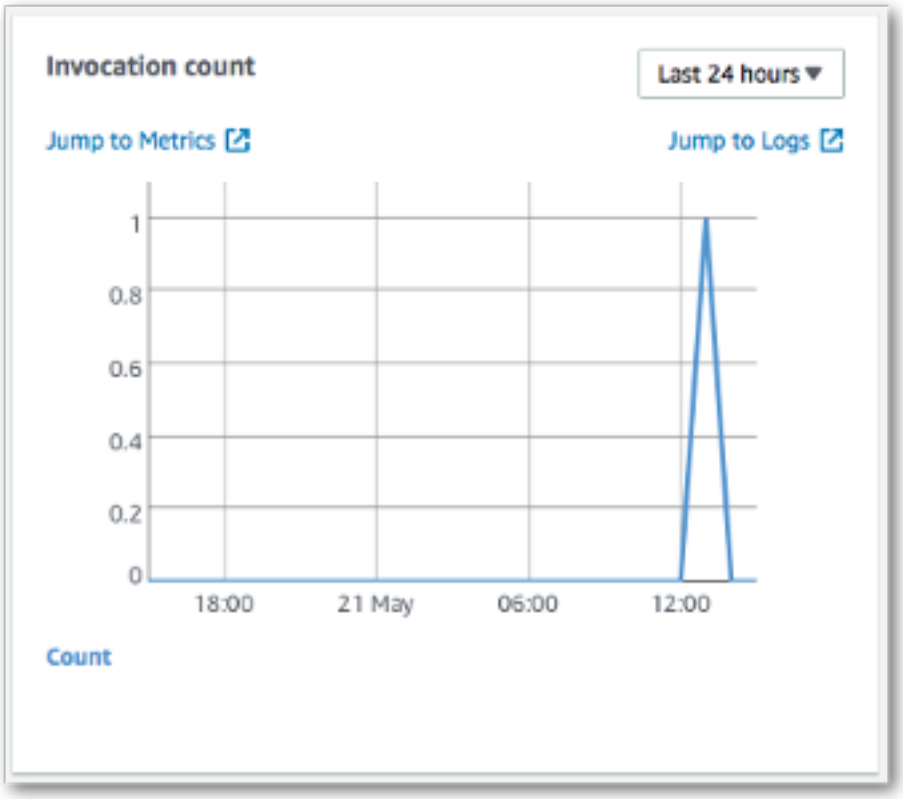
步骤 6：验证您的消息传输

当 Amazon SNS 主题收到一条消息时，它会通过将其发送到两个订阅的 Lambda 函数来分散消息。当这些函数收到消息时，它们会将事件记录到 CloudWatch 日志中。要验证您的消息传送是否成功，请检查函数是否已调用，并检查 CloudWatch 日志是否已更新。

验证是否已调用 Lambda 函数

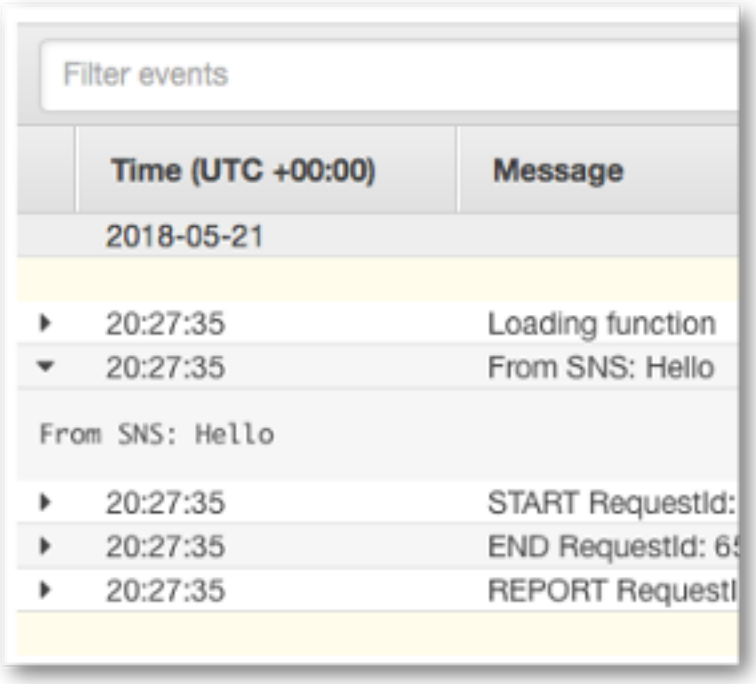
1. 打开 AWS Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 在函数页面上，选择 VPCE-教程-Lambda-1。
3. 选择监控。
4. 检查 Invocation count (调用计数) 图表。此图显示了已运行 Lambda 函数的次数。

   调用计数与您向主题发布消息的次数匹配。



验证 CloudWatch 日志是否已更新

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航菜单中，选择 Logs (日志)。
3. 检查由 Lambda 函数写入的日志：
   a. 选择 /aws/lambda/-Tutorial-Lambda-1/ VPCE 日志组。
   b. 选择日志流。
   c. 检查日志是否包含条目 `From SNS: Hello`。



d. 选择控制台顶部的 Log Groups (日志组) 以返回 Log Groups (日志组) 页面。然后，对 /aws/lambda/-Tutorial-Lambda-2/ 日志组VPCE重复上述步骤。

恭喜您！通过向 Amazon SNS 添加终端节点VPC，您可以从管理的网络内向主题发布消息VPC。消息是私下发布的，不会对公共 Internet 公开。

步骤 7：清除

除非您想要保留您创建的资源，否则可立即将其删除。通过删除不再使用的 AWS 资源，可以防止对您的资源产生不必要的费用 AWS 账户。

首先，使用 Amazon VPC 控制台删除您的VPC终端节点。然后，通过在 AWS CloudFormation 控制台中删除堆栈来删除您创建的其他资源。当您删除堆栈时， AWS CloudFormation 会将该堆栈的资源从中移除 AWS 账户。

删除您的VPC终端节点

1. 打开亚马逊VPC控制台，网址为https://console.aws.amazon.com/vpc/。
2. 在左侧导航菜单中，选择 Endpoints (终端节点)。
3. 选择您创建的终端节点。
4. 选择 Actions (操作)，然后选择 Delete Endpoint (终端节点)。
5. 在 Delete Endpoint (删除终端节点) 窗口中，选择 Yes, Delete (是，删除)。

终端节点状态将变为 deleting (正在删除)。删除操作完成后，终端节点将从页面中删除。

删除您的 AWS CloudFormation 堆栈

1. 在 https://console.aws.amazon.com/cloudformation 上打开 AWS CloudFormation 控制台。
2. 选择堆栈 VPCE-教程堆栈。
3. 选择 Actions) (操作) ，然后选择 Delete Stack (删除堆栈) 。
4. 在 Delete Stack (删除堆栈) 窗口中，选择 Yes, Delete (是，删除)。

堆栈状态更改为 DELETE_IN_ PROGRESS。删除操作完成后，堆栈将从页面中删除。

相关资源

有关详细信息，请参阅以下资源：

- AWS 安全博客：使用保护发布到 Amazon SNS 的消息 AWS PrivateLink
- 什么是亚马逊VPC？
- VPC端点
- 什么是亚马逊EC2？
- AWS CloudFormation 概念

通过消息数据保护增强 Amazon 的SNS安全

- 消息数据保护是 Amazon 的一项功能，SNS用于定义您自己的规则和政策，以审计和控制动态数据（而不是静态数据）的内容。
- Message Data Protection 为以消息为中心的企业应用程序提供监管、合规和审计服务，因此 Amazon SNS 主题所有者可以控制数据的入口和出口，并且可以跟踪和记录内容流。
- 您可以编写基于负载的监管规则，以阻止未经授权的负载内容进入您的消息流。
- 您可以向单独订阅者授予不同的内容访问权限，并审计整个内容流流程。

Amazon 中的身份和访问管理 SNS

访问 Amazon SNS 需要 AWS 可用于验证您的请求的凭证。这些证书必须具有访问 AWS 资源（例如 Amazon SNS 主题和消息）的权限。以下各节详细介绍了如何使用 AWS Identity and Access Management (IAM) 和 Amazon 通过控制对资源的访问SNS来保护您的资源。

AWS Identity and Access Management (IAM) AWS 服务 可以帮助管理员安全地控制对 AWS 资源的访问权限。IAM管理员控制谁可以通过身份验证（登录）和授权（拥有权限）使用 Amazon SNS 资源。IAM无需支付额外费用即可使用。 AWS 服务

受众

您使用 AWS Identity and Access Management (IAM) 的方式会有所不同，具体取决于您在 Amazon 上所做的工作SNS。

服务用户 — 如果您使用 Amazon SNS 服务完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多的 Amazon SNS 功能来完成工作时，您可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amazon 中的某项功能SNS，请参阅Amazon Simple Notification Service 身份和访问故障排查。

服务管理员 — 如果您负责公司的亚马逊SNS资源，则可能拥有对亚马逊的完全访问权限SNS。您的工作是确定您的服务用户应该访问哪些亚马逊SNS功能和资源。然后，您必须向IAM管理员提交更改服务用户权限的请求。查看此页面上的信息以了解的基本概念IAM。要详细了解贵公司如何IAM与 Amazon 搭配使用SNS，请参阅亚马逊是如何SNS与之合作的 IAM。

IAM管理员 — 如果您是IAM管理员，则可能需要详细了解如何编写策略来管理对 Amazon 的访问权限SNS。要查看您可以在中使用的SNS基于身份的 Amazon 策略示例IAM，请参阅。适用于 Amazon Simple Notification Service 的基于身份的策略示例

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须 AWS 账户 以 root 用户、用户身份或通过扮IAM演角色进行IAM身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM身份中心）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员之前使用IAM角色设置了联合身份。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM用户指南》中的对 AWS API请求进行签名。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅用户指南中的多重身份验证和AWS IAM Identity Center 用户指南 AWS中的使用多因素身份验证 (MFA)。IAM

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以 root 用户身份登录的任务的完整列表，请参阅《用户指南》中的 “需要根用户凭据的IAM任务”。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 Ident IAM ity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户和应用程序中使用。有关IAM身份中心的信息，请参阅什么是IAM身份中心？ 在《AWS IAM Identity Center 用户指南》中。

IAM 用户和组

IAM用户是您内部 AWS 账户 对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时证书，而不是创建拥有密码和访问密钥等长期凭证的IAM用户。但是，如果您有需要IAM用户长期凭证的特定用例，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM用户指南》中的针对需要长期凭证的用例定期轮换访问密钥。

IAM群组是指定IAM用户集合的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并授予该群组管理IAM资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM用户指南》中的何时创建IAM用户（而不是角色）。

IAM角色

IAM角色是您内部具有特定权限 AWS 账户 的身份。它与IAM用户类似，但与特定人员无关。您可以通过切换IAM角色 AWS Management Console 来临时担任中的角色。您可以通过调用 AWS CLI 或 AWS API操作或使用自定义操作来代入角色URL。有关使用角色的方法的更多信息，请参阅《IAM用户指南》中的使用IAM角色。

IAM具有临时证书的角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为了控制您的身份在进行身份验证后可以访问的内容，Ident IAM ity Center 会将权限集关联到中的IAM角色。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。
- 临时IAM用户权限-IAM 用户或角色可以代入一个IAM角色，为特定任务临时获得不同的权限。
- 跨账户访问-您可以使用IAM角色允许其他账户中的某人（受信任的委托人）访问您账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序EC2或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS)-当您使用IAM用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用调用委托人的权限 AWS 服务以及 AWS 服务 向下游服务发出请求的请求。FAS只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出请求。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。
  - 服务角色-服务IAM角色是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅《IAM用户指南》 AWS 服务中的创建角色以向委派权限。

- 服务相关角色-服务相关角色是一种与服务相关联的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon 上运行的应用程序 EC2 — 您可以使用IAM角色管理在EC2实例上运行并发出 AWS CLI 或 AWS API请求的应用程序的临时证书。这比在EC2实例中存储访问密钥更可取。要为EC2实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例配置文件包含该角色，并允许在EC2实例上运行的程序获得临时证书。有关更多信息，请参阅IAM用户指南中的使用IAM角色向在 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用IAM角色还是使用IAM用户，请参阅《用户指南》中的何时创建IAM角色（而不是IAM用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以JSON文档的 AWS 形式存储在中。有关JSON策略文档结构和内容的更多信息，请参阅《IAM用户指南》中的JSON策略概述。

管理员可以使用 AWS JSON策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入角色。

IAM无论您使用何种方法执行操作，策略都会定义该操作的权限。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或获取角色信息 AWS API。

基于身份的策略

基于身份的策略是可以附加到身份（例如IAM用户、用户组或角色）的JSON权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅IAM用户指南中的创建IAM策略。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括

AWS 托管策略和客户托管策略。要了解如何在托管策略或内联策略之间进行选择，请参阅《IAM用户指南》中的在托管策略和内联策略之间进行选择。

基于资源的策略

基于资源的JSON策略是您附加到资源的策略文档。基于资源的策略的示例包括IAM角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略IAM中使用 AWS 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACLs与基于资源的策略类似，尽管它们不使用JSON策略文档格式。

Amazon S3 AWS WAF、和亚马逊VPC就是支持的服务示例ACLs。要了解更多信息ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界-权限边界是一项高级功能，您可以在其中设置基于身份的策略可以向IAM实体（IAM用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM用户指南》中的IAM实体的权限边界。
- 服务控制策略 (SCPs)-SCPs 是为中的组织或组织单位 (OU) 指定最大权限的JSON策略 AWS Organizations。 AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。SCP限制了成员账户中实体的权限，包括每个 AWS 账户 root 用户。有关 Organization SCPs s 和的更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅IAM用户指南中的策略评估逻辑。

访问控制

Amazon SNS 拥有自己的基于资源的权限系统，该系统使用与 AWS Identity and Access Management (IAM) 策略相同的语言编写的策略。这意味着您可以通过Amazon的SNSIAM政策和政策实现类似的目标。

> **Note**
>
> 重要的是要明白，所有人 AWS 账户 都可以将其权限委托给其账户下的用户。跨账户访问允许您共享对 AWS 资源的访问，而不需要管理其他用户。有关使用跨账户访问的信息，请参阅《IAM用户指南》中的 “启用跨账户访问权限”。

在 Amazon 中管理访问权限概述 SNS

本部分介绍了使用访问策略语言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访问策略语言一起使用的一般过程以及如何评估策略。

主题

- Amazon SNS 访问控制用例
- 亚马逊SNS访问策略的关键概念
- Amazon SNS 访问控制架构概述
- 在 Amazon 中使用访问策略语言 SNS
- 评估逻辑
- Amazon SNS 访问控制的示例案例

Amazon SNS 访问控制用例

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。

- 您想向其他人授予 AWS 账户 特定类型的主题操作（例如 “发布”）。有关更多信息，请参阅 授予对主题的 AWS 账户 访问权限。

- 您想将主题的订阅限制为仅限HTTPS协议。有关更多信息，请参阅 将订阅限制为 HTTPS。
- 您想允许亚马逊SNS向您的亚马逊SQS队列发布消息。有关更多信息，请参阅 将消息发布到 Amazon SQS 队列。

亚马逊SNS访问策略的关键概念

以下章节介绍了您需要了解的概念，以便使用访问策略语言。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。

主题

- 权限
- 语句
- Policy
- Issuer
- 主体
- 操作
- 资源
- 条件与密钥
- 请求者
- 评估
- 效果
- 默认拒绝
- 允许
- 显式拒绝

权限

权限是指允许或不允许访问某种特殊资源的概念。权限基本遵循这种形式：“如果 D 适用时，那么 A 被允许或未被允许向 C 执行 B”。例如，只要 Jane (A) 使用HTTP协议 (D)，她就有权将 (B) 发布到 topicA (C)。每当 Jane 向 TopicA 发布信息时，服务器会检查她是否有权限或该请求是否满足权限中所规定的条件。

语句

语句是指对用访问策略语言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。

Policy

策略是一个文档（使用访问策略语言编写），充当存放一个或多个语句的容器。例如，策略中可能包含两个语句：一个语句规定 Jane 可以使用电子邮件协议进行订阅，另一个语句规定 Bob 不能向主题 A 发布消息。如下图所示，等效的场景将具有两个策略，一个策略规定 Jane 可以使用电子邮件协议进行订阅，另一个策略规定 Bob 不能向主题 A 发布消息。

政策文件中只允许使用ASCII字符。你可以利用aws:SourceAccount和解决aws:SourceOwner需要插入其他包含非ASCII字符 AWS ARNs的服务的场景。请参阅 aws:SourceAccount 与 aws:SourceOwner 之间的差异。

Issuer

发布者是指编写策略以授予资源权限的人。发行者（根据定义）始终是资源所有者。 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，则在 John 提交他为授予该资源权限而编写的策略时，将 AWS 验证他的身份。

主体

委托人是指在策略中获取权限的个人和多个人。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。在策略中，您可将委托人设置为“任何人”（即，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，例如请求者的 IP 地址。

操作

操作是委托人可以执行的活动。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，操作只是请求中的操作 AWS。例如，Jane SNS 使用向亚马逊发送请求`Action=Subscribe`。在一个策略中您可指定一个或多个操作。

资源

资源是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

条件是任何有关权限的限制条件和具体内容。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 地址必须属于特定CIDR范围）

一个密钥是设置访问限制指定的特性。例如，访问日期和时间。

您需使用条件和密钥一起明确说明限制。下列示例可帮助您以最简单的方式了解如何实际实施限制：若您要在 2010 年 5 月 30 日之前限制访问，则使用名为 `DateLessThan` 的条件。您可以使用名为 `aws:CurrentTime` 的密钥并将其设置为 `2010-05-30T00:00:00Z`。 AWS 定义您可以使用的条件和键。 AWS 服务本身（例如，Amazon SQS 或 AmazonSNS）也可能定义特定于服务的密钥。有关更多信息，请参阅 Amazon SNS API 权限：操作和资源参考。

请求者

请求者指向一个 AWS 服务发出请求并要求访问某个特定资源的人。请求者向 AWS 发送的请求基本表述为：“在满足 D 的情况下，您是否允许我对 C 执行 B？”

评估

评估是 AWS 服务根据适用的策略确定是否应拒绝或允许传入请求的流程。有关评估逻辑的信息，请查见评估逻辑。

效果

效果是指在评估期间您希望一个策略语句返回的结果。当您在一个策略中编写语句时，您需指定该值，可能值为拒绝和允许。

例如，您可以编写一个策略，并声明拒绝所有来自南极洲地区的请求（效果相当于：如果请求是使用为南极洲配置的 IP 地址发出的，则拒绝该请求）。同样地，您可以编写一个策略，其中声明允许所有并非来自南极洲地区的请求（效果=如果请求不是来自南极洲地区，则允许）。虽然这两个语句看似执行相同的操作，但是在访问策略语言逻辑上，它们是不同的。有关更多信息，请参阅 评估逻辑。

尽管仅可指定两个效果值（“允许”或“拒绝”），但在执行策略评估时，却可能产生三种不同的结果，即：默认拒绝、允许或显式拒绝。有关更多详细，参见以下概念和 评估逻辑。

默认拒绝

默认拒绝是从一个策略中没有允许或显式拒绝的默认结果。

允许

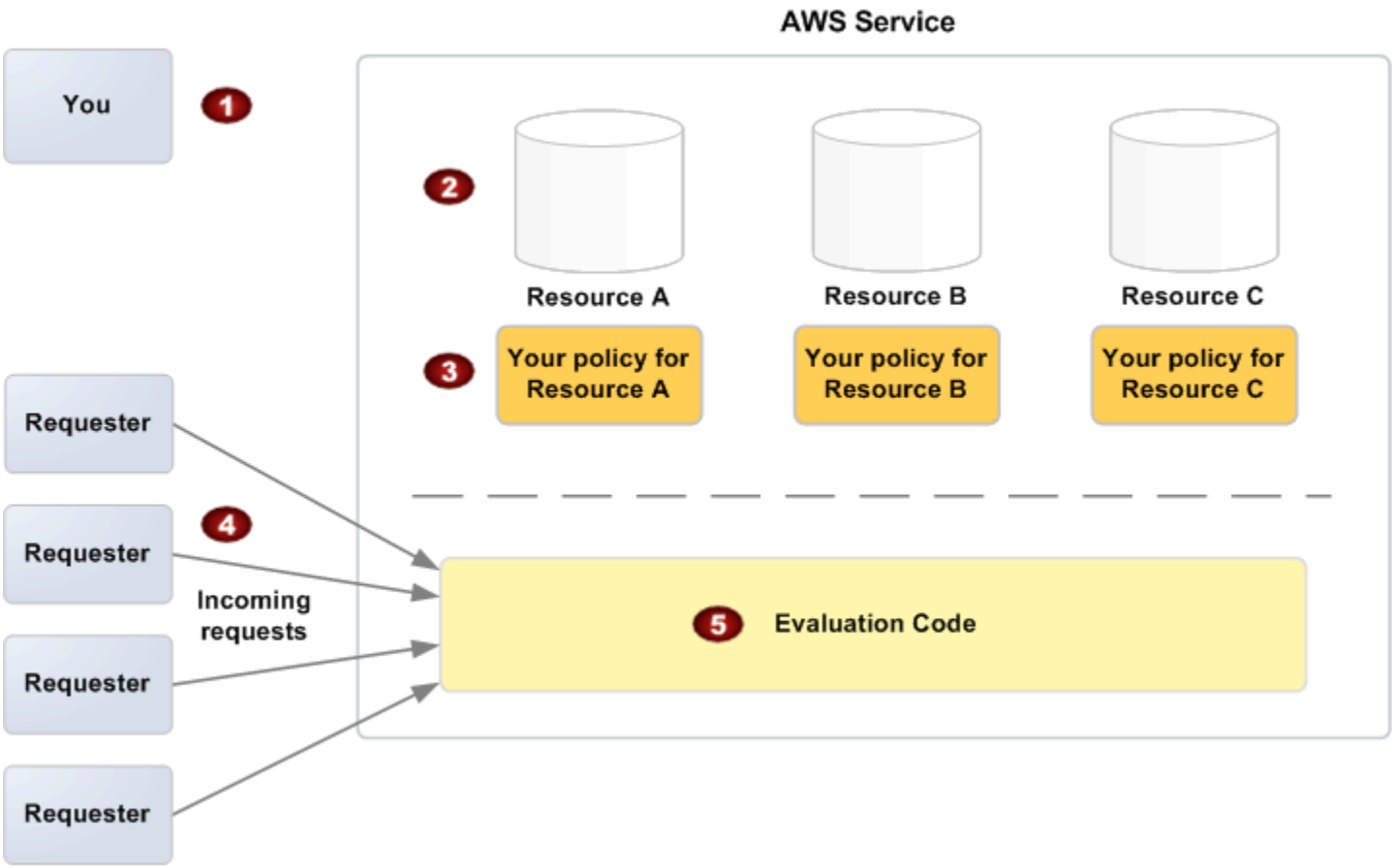
假设任何声明的条件已满足，效果=允许的语句会产生允许。例如：若在 2010 年 4 月 30 日下午 1:00 之前收到请求，则请求将获得允许。“允许”可置换所有“默认拒绝”，但无法置换“显式拒绝”。

显式拒绝

假设任何声明的条件已满足，效果=拒绝的语句会产生显式拒绝。例如：拒绝来自南极洲的所有请求。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。

Amazon SNS 访问控制架构概述

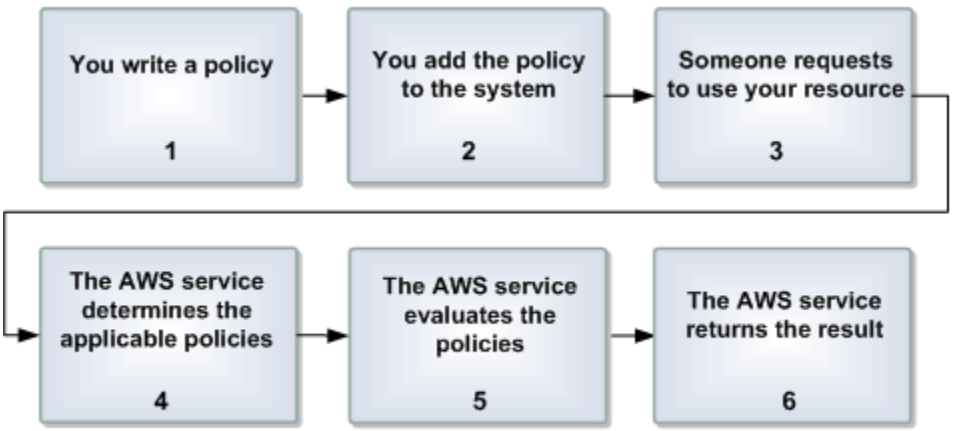
下列图和表格介绍了为您提供资源访问控制的相互作用的主要组成部分。



1	您，资源所有者。
2	您的资源（包含在 AWS 服务中；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。该 AWS 服务本身提供了一个供 API您上传和管理策略的工具。
4	请求者及其对 AWS 服务的传入请求。
5	访问策略语言评估代码。 这是 AWS 服务中的一组代码，用于根据适用的策略评估传入的请求，并确定是否允许请求者访问资源。有关产品如何做出决定的信息，请参见 评估逻辑。

在 Amazon 中使用访问策略语言 SNS

下面的图表介绍了访问控制与访问策略语言协作的一般过程。



将访问控制与访问策略语言一起使用的过程

1	为您的资源编写一个策略。 例如，您编写了一份策略来指定您的 Amazon SNS 主题的权限。
2	您将您的保单上传到 AWS。 该 AWS 服务本身提供了一个供API您上传政策的工具。例如，您可以使用 Amazon SNS `SetTopicAttributes` 操作上传针对特定亚马逊SNS主题的政策。
3	某人向您发出使用您的资源的请求。 例如，用户向 Amazon 发送请求SNS以使用您的一个主题。
4	该 AWS 服务决定哪些策略适用于该请求。 例如，Amazon 会SNS查看所有可用的亚马逊SNS政策，并确定哪些政策适用（基于资源是什么、申请者是谁等）。
5	该 AWS 服务对策略进行评估。 例如，Amazon SNS 会评估政策并确定是否允许请求者使用您的主题。有关决策逻辑的信息，请参见评估逻辑。
6	该 AWS 服务要么拒绝请求，要么继续处理请求。

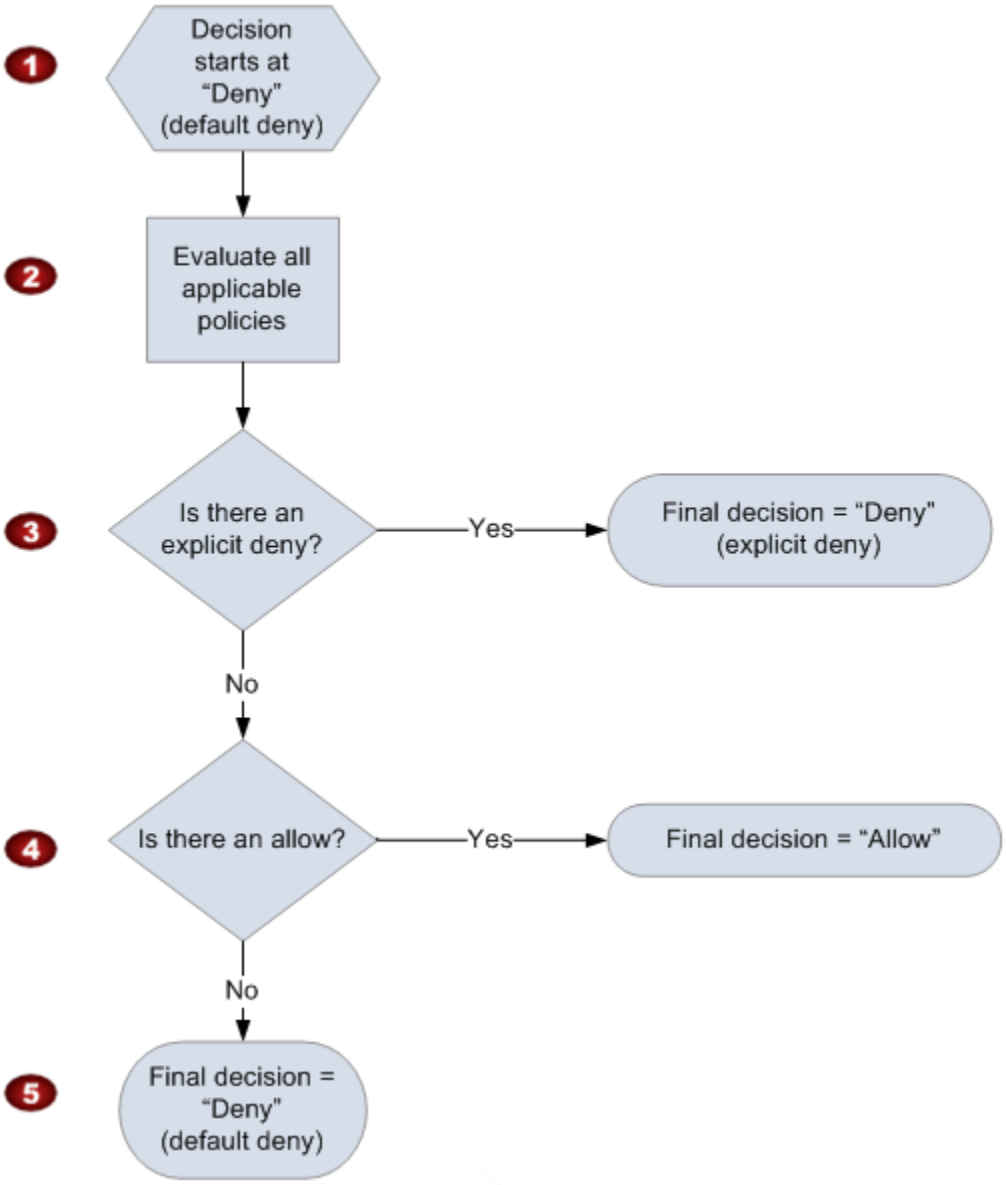
例如，根据策略评估结果，服务将返回一个“访问被拒绝”的错误信息给请求者，或继续处理该请求。

评估逻辑

评价期间的目的是为了确定应该允许还是拒绝授予请求。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。 即使仅找到一处，执行代码也会发回“拒绝”的决定，并结束处理流程（此为“显式拒绝”；有关更多信息，请参见 显式拒绝）。

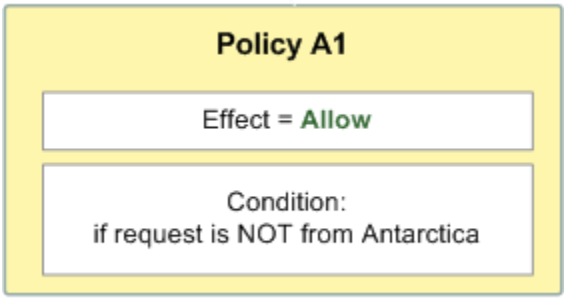
4	如果没有找到显式拒绝，那么执行代码将寻找适用于请求的任何“允许”指令。 如果它还是找到了一个，那么执行代码将返回一个“允许”决定，且整个过程完成（服务将继续处理该请求）。
5	如果没有找到允许，那么最终的决定将是“拒绝”，因为没有显式拒绝或允许，所以这将被视为是一个默认拒绝（有关更多信息，请参见默认拒绝）。

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 AmazonSNS，但该主题的政策根本没有提及该用户的 AWS 账户 政策，则该策略会导致默认拒绝。

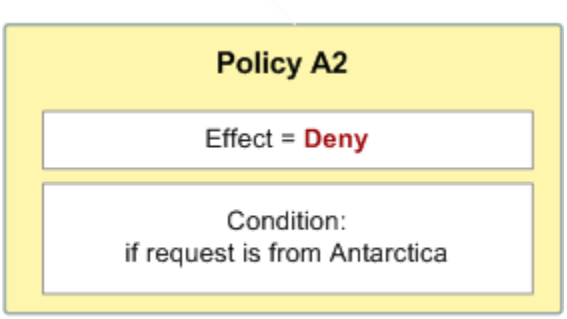
如果一个语句中的某个条件未被满足，那么策略将产生一个默认拒绝。如果语句中的所有条件都满足，那么根据策略中的效果元素的值，策略或许会产生允许，或许会产生显式拒绝。如果一个条件未被满足，策略没有指定如何处理，那么在那种情况下默认值将产生一个默认拒绝。

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。



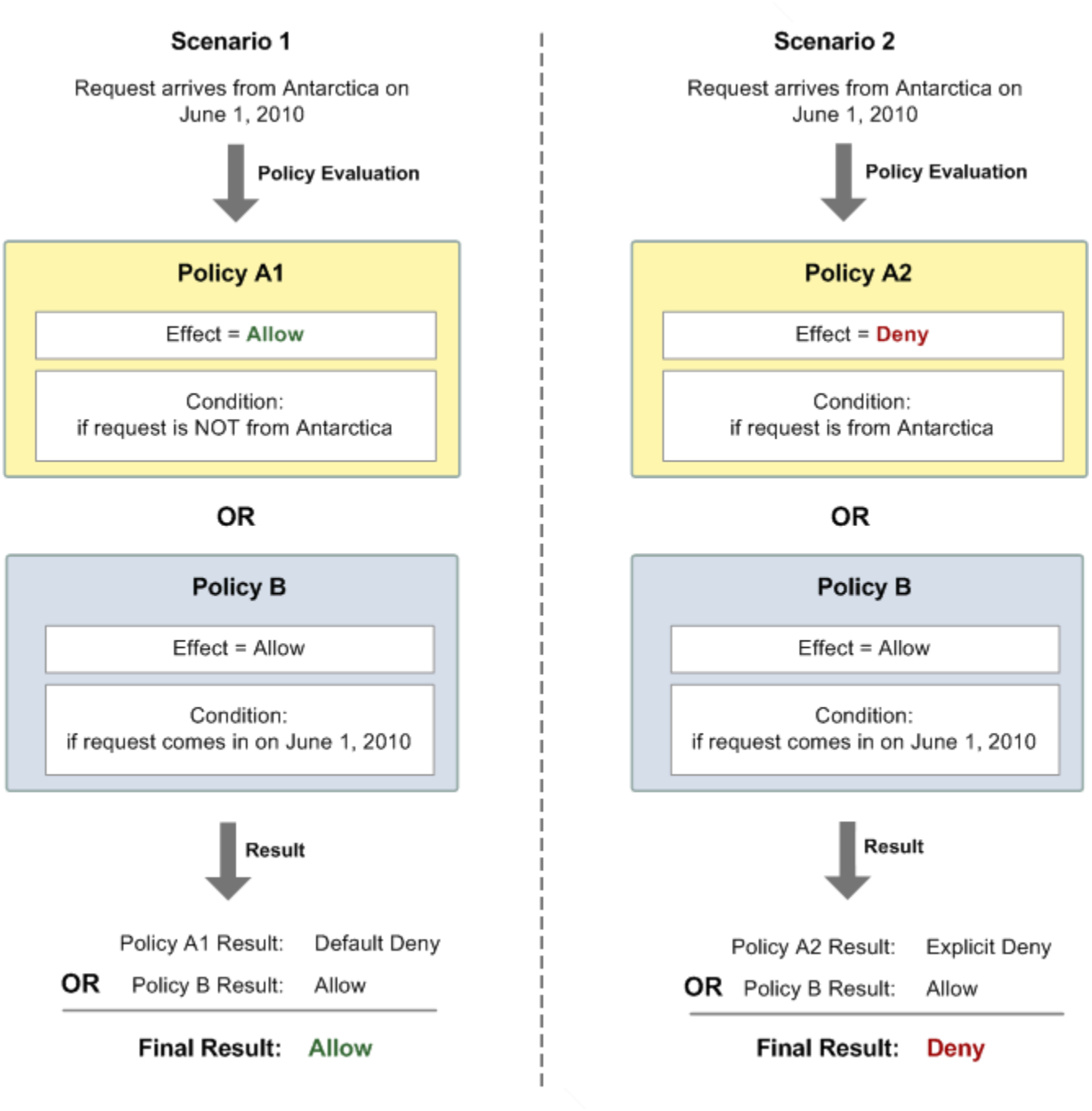
如果某人从美国发出请求，那么条件已经满足 (该请求不是来自南极洲)。因此，该请求将被允许。但是，如果某人从南极洲地区发出请求，那么条件未满足，因此策略结果将是默认拒绝。

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，但显式拒绝就不能。例如，假设有另一个策略，允许在 2010 年 6 月 1 日到达的请求。那么，与限制从南极洲访问的策略相结合，该策略将如何对总体结果产生影响？ 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 日从南极洲区域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。Policy B 返回“允许”结果，因为该策略（依照定义）允许在 2010 年 6 月 1 日发送请求。Policy B 返回的“允许”结果将置换 Policy A1 的“默认拒绝”结果，因此，请求获得允许。

在方案 2 中，策略 A2 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。

Amazon SNS 访问控制的示例案例

本节描述了针对访问控制的几个典型使用案例示例。

主题

授予对主题的 AWS 账户 访问权限

假设您在 Amazon 中有一个主题SNS，并且您想允许一个或多个 AWS 账户 主题对该主题执行特定操作，例如发布消息。您可以使用 Amazon SNS API 操作来完成此操作AddPermission。

该AddPermission操作允许您指定主题、列表 AWS 账户 IDs、操作列表和标签。SNS然后，Amazon 会自动生成新的政策声明，并将其添加到该主题的访问控制策略中。您无需自己撰写政策声明，亚马逊会为您SNS处理。如果您以后需要删除该策略，则可以通过致电RemovePermission并提供您在添加权限时使用的标签来实现。

例如，如果您调用 AddPermission arn: aws: sns: us-east-2:444455556666:，MyTopic指定 ID AWS 账户 1111-2222-3333 Publish、操作和标签，亚马逊将生成以下策略声明并将其插入到该主题的访问控制策略中：grant-1234-publishSNS

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
```

```
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

添加此声明后， AWS 账户 1111-2222-3333 将有权向该主题发布消息。

其他信息：

- 自定义策略管理：虽然AddPermission可以方便地授予权限，但对于更复杂的场景，例如添加条件或向特定IAM角色或服务授予权限，手动管理主题的访问控制策略通常很有用。为此，您可以使用SetTopicAttributesAPI直接更新策略属性。
- 安全最佳实践：在授予权限时要谨慎行事，确保只有受信任的实体 AWS 账户 或实体才能访问您的 Amazon SNS 主题。定期审查和审核您的主题所附的政策，以维护安全性。
- 政策限制：请记住，Amazon SNS 政策的规模和复杂性有限制。如果您需要添加许多权限或复杂条件，请确保您的策略保持在这些限制范围内。

将订阅限制为 HTTPS

要将您的 Amazon SNS 主题的通知传送协议限制为HTTPS，您必须创建自定义策略。Amazon 中的AddPermission操作SNS不允许您在授予主题访问权限时指定协议限制。因此，您需要手动编写强制执行此限制的策略，然后使用SetTopicAttributes操作将该策略应用于您的主题。

您可以通过以下方式创建政策，将订阅限制为HTTPS：

1. 撰写政策。该策略必须指定您要授予访问权限的 AWS 账户 ID，并强制执行仅允许HTTPS订阅的条件。以下是一个策略示例，它授予 AWS 账户 ID 1111-2222-3333 订阅该主题的权限，但前提是使用的协议是。HTTPS

```
{
  "Statement": [{
    "Sid": "Statement1",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Subscribe"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringEquals": {
        "sns:Protocol": "https"
```

```
      }
    }
  }]
}
```

2. 应用政策。使用 Amazon 中的SetTopicAttributes操作SNSAPI将此政策应用于您的主题。将主题的Policy属性设置为您创建的JSON策略。

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:444455556666:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

其他信息：

- 自定义访问控制。这种方法允许您实施更精细的访问控制，例如限制订阅协议，而光靠AddPermission操作是不可能的。自定义策略为需要特定条件（例如协议强制执行或 IP 地址限制）的场景提供了灵活性。
- 安全最佳实践。限制订阅HTTPS可以确保传输中的数据经过加密，从而增强通知的安全性。定期查看您的主题政策，确保它们符合您的安全和合规性要求。
- 政策测试。在生产环境中应用策略之前，请在开发环境中对其进行测试，以确保其行为符合预期。这有助于防止意外访问问题或意外限制。

将消息发布到 Amazon SQS 队列

要将来自您的亚马逊SNS主题的消息发布到亚马逊SQS队列，您需要在亚马逊SQS队列上配置正确的权限。虽然亚马逊SNS和亚马逊都SQS使用 AWS访问控制策略语言，但您必须在亚马逊SQS队列上明确设置策略，以允许从亚马逊SNS主题发送消息。

您可以通过使用SetQueueAttributes操作将自定义策略应用于 Amazon SQS 队列来实现此目的。与亚马逊不同SNS，亚马逊SQS不支持创建带有条件的政策声明的AddPermission操作。因此，您必须手动编写策略。

以下是授予亚马逊向您的队列发送消息的SNS权限的亚马逊SQS政策示例。请注意，此政策与亚马逊SQS队列相关联，而不是与亚马逊SNS主题相关联。指定的操作是亚马逊SQS操作，资源是队列的亚马逊资源名称 (ARN)。您可以使用GetQueueAttributes操作检索ARN队列。

```
{
```

```
  "Statement": [{
    "Sid": "Allow-SNS-SendMessage",
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": ["sqs:SendMessage"],
    "Resource": "arn:aws:sqs:us-east-2:444455556666:MyQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:444455556666:MyTopic"
      }
    }
  }]
}
```

此策略使用aws:SourceArn条件根据所发送消息的来源限制对SQS队列的访问权限。这样可以确保只允许将来自指定SNS主题（在本例中为 arn: aws: sns: us-east-2:444455556666: ）的消息传送到队列。MyTopic

其他信息：

- 队列ARN。确保使用GetQueueAttributes操作检索正确ARN的 Amazon SQS 队列。这ARN对于设置正确的权限至关重要。
- 安全最佳实践。设置策略时，请始终遵循最低权限原则。仅向亚马逊SNS主题授予与亚马逊SQS队列进行交互的必要权限，并定期查看您的政策，确保政策的有效 up-to-date 性和安全性
- Amazon 中的默认政策SNS。与某些误解相反，Amazon SNS 不会自动授予允许其他人 AWS 服务访问新创建主题的默认策略。您必须明确定义并附加策略以控制对您的 Amazon SNS 主题的访问权限。
- 测试和验证。设置策略后，通过向亚马逊SNS主题发布消息并验证消息是否已成功传送到亚马逊SQS队列来测试集成。这有助于确认策略的配置是否正确。

允许 Amazon S3 事件通知发布到主题

要允许另一个 AWS 账户 Amazon S3 存储桶向您的亚马逊SNS主题发布事件通知，您需要相应地配置该主题的访问策略。这包括编写自定义策略，向特定的 Amazon S3 服务授予权限， AWS 账户 然后将此策略应用于您的 Amazon SNS 主题。

您可以通过以下方式进行设置：

1. 撰写政策。该政策应授予 Amazon S3 服务 (s3.amazonaws.com) 发布您的亚马逊SNS主题所需的权限。您将使用SourceAccount条件来确保只有拥有 Amazon S3 存储桶的指定 AWS 账户人员才能向您的主题发布通知。

   下面是一个策略示例：

   ```
   {
     "Statement": [{
       "Effect": "Allow",
        "Principal": {
         "Service": "s3.amazonaws.com"
       },
       "Action": "sns:Publish",
       "Resource": "arn:aws:sns:us-east-2:111122223333:MyTopic",
       "Condition": {
         "StringEquals": {
           "AWS:SourceAccount": "444455556666"
         }
       }
     }]
   }
   ```

   - 话题所有者 — 111122223333 是拥有亚马逊话题 AWS 账户 的 ID。SNS
   - 亚马逊 S3 存储桶所有者 — 444455556666 是 AWS 账户 拥有发送通知的亚马逊 S3 存储桶的 ID。
2. 应用政策。使用SetTopicAttributes操作针对您的 Amazon SNS 主题设置此政策。这将更新主题的访问控制，使其包含您的自定义策略中指定的权限。

   ```
   snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
           .topicArn("arn:aws:sns:us-east-2:111122223333:MyTopic")
           .attributeName("Policy")
           .attributeValue(jsonPolicyString)  // The JSON policy as a string
           .build());
   ```

其他信息：

- 使用**SourceAccount**条件。该SourceAccount条件可确保只有源自指定 AWS 账户 （在本例中为 444455556666）的事件才能触发亚马逊主题。SNS这是一项安全措施，可防止未经授权的账户向您的主题发送通知。

- 其他服务支持**SourceAccount**。以下服务支持该SourceAccount条件。当您想要根据原始账户限制对您的 Amazon SNS 主题的访问时，使用此条件至关重要。
  - 亚马逊API网关
  - Amazon CloudWatch
  - Amazon DevOps Guru
  - Amazon ElastiCache
  - Amazon EventBridge
  - Amazon GameLift
  - 亚马逊 Pinpoint SMS 和语音 API
  - Amazon RDS
  - Amazon Redshift
  - Amazon S3 Glacier
  - Amazon SES
  - Amazon Simple Storage Service
  - AWS CodeCommit
  - AWS Directory Service
  - AWS Lambda
  - AWS Systems Manager Incident Manager
- 测试和验证。应用策略后，通过在 Amazon S3 存储桶中触发事件并确认该事件已成功发布到您的 Amazon SNS 主题来测试设置。这将有助于确保您的策略配置正确。
- 安全最佳实践。定期审查和审核您的 Amazon SNS 主题政策，确保它们符合您的安全要求。仅限可信账户和服务访问权限对于维护安全运营至关重要。

允许 Amazon SES 向其他账户拥有的主题发布内容

您可以 AWS 服务 允许其他人发布他人拥有的主题 AWS 账户。假设您登录了 111122223333 账户，打开了SES亚马逊，并创建了一封电子邮件。要向 444455556666 账户拥有的亚马逊SNS主题发布有关此电子邮件的通知，您需要创建如下政策。为此，您需要提供有关委托人（其他服务）和每个资源的所有权的信息。该Resource声明提供了主题ARN，其中包括主题所有者的账户 ID 444455556666。"aws:SourceOwner": "111122223333" 语句指定您的账户拥有该电子邮件。

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
```

```
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
      "Condition": {
        "StringEquals": {
          "aws:SourceOwner": "111122223333"
        }
      }
    }
  ]
}
```

向 Amazon 发布活动时SNS，以下服务支持`aws:SourceOwner`：

- 亚马逊API网关
- Amazon CloudWatch
- Amazon DevOps Guru
- Amazon ElastiCache
- Amazon GameLift
- 亚马逊 Pinpoint SMS 和语音 API
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

aws:SourceAccount 与 aws:SourceOwner

> **Important**
>
> aws:SourceOwner已弃用，新服务SNS只能通过aws:SourceArn和aws:SourceAccount与 Amazon 集成。对于目前支持的现有服务，Amazon SNS 仍保持向后兼容性aws:SourceOwner。

aws:SourceAccount和aws:SourceOwner条件键均由某些人在发布到 Amazon SNS 主题 AWS 服务 时设置。如果支持，则该值将是服务代表其发布数据的 12 位 AWS 账户 ID。一些服务支持其中一项，一些服务支持另一项。

- 允许 Amazon S3 事件通知发布到主题请参阅，了解 Amazon S3 通知的使用方式aws:SourceAccount以及支持该条件的 AWS 服务列表。
- 允许 Amazon SES 向其他账户拥有的主题发布内容有关Amazon的SES使用方式aws:SourceOwner以及支持该条件的 AWS 服务列表，请参阅。

允许组织中的账户 AWS Organizations 向其他账户中的主题发布内容

该 AWS Organizations 服务可帮助您集中管理账单，控制访问权限和安全性，并在整个系统中共享资源 AWS 账户。

您可以在 Organizations 控制台中找到您的组织 ID。有关更多信息，请参阅从管理账户查看组织的详细信息。

在此示例中，组织 AWS 账户 中的任何成员都myOrgId可以在账户MyTopic中向 Amazon 发布SNS主题444455556666。该策略使用 aws:PrincipalOrgID 全局条件键检查组织 ID 值。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
}
```

允许将任何 CloudWatch 警报发布到其他账户中的某个主题

在这种情况下，允许将账户111122223333中的任何 CloudWatch 警报发布到账户中的亚马逊SNS主题444455556666。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudwatch:us-
east-2:111122223333:alarm:*"
                }
            }
        }
    ]
}
```

限制仅从特定VPC终端节点发布到 Amazon SNS 主题

在这种情况下，账户 44455556666 中的主题只能从带有 ID 的终端节点发布。VPC vpce-1ab2c34d

```
{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "SNS:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
```

```
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
  }]
}
```

亚马逊是如何SNS与之合作的 IAM

在使用IAM管理亚马逊访问权限之前SNS，请先了解亚马逊有哪些IAM功能可供使用SNS。

IAM您可以在 Amazon 简单通知服务中使用的功能

IAM特征	亚马逊SNS支持
基于身份的策略	是
基于资源的策略	是
策略操作	是
策略资源	是
策略条件键（特定于服务）	是
ACLs	不支持
ABAC（策略中的标签）	部分
临时凭证	是
主体权限	是
服务角色	是
服务相关角色	否

要全面了解 Amazon SNS 和其他 AWS 服务如何使用大多数IAM功能，请参阅《IAM用户指南》IAM中与之配合使用的AWS 服务。

针对亚马逊的政策行动 SNS

支持策略操作：是

管理员可以使用 AWS JSON策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

JSON策略`Action`元素描述了可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API操作同名。也有一些例外，例如没有匹配API操作的仅限权限的操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看亚马逊SNS操作列表，请参阅《服务授权参考》中的 “亚马逊简单通知服务定义的资源”。

Amazon 中的策略操作在操作前SNS使用以下前缀：

```
sns
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

要查看 Amazon SNS 基于身份的政策示例，请参阅。适用于 Amazon Simple Notification Service 的基于身份的策略示例

Amazon 的政策资源 SNS

支持策略资源：是

管理员可以使用 AWS JSON策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

`Resource`JSON策略元素指定要应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。最佳做法是，使用资源的 Amazon 资源名称 (ARN) 来指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看亚马逊SNS资源类型及其列表ARNs，请参阅《服务授权参考》中的 “亚马逊简单通知服务定义的操作”。要了解您可以为每种资源指定哪些操作，请参阅 Amazon 简单通知服务定义的资源。ARN

要查看 Amazon SNS 基于身份的政策示例，请参阅。适用于 Amazon Simple Notification Service 的基于身份的策略示例

Amazon 的政策条件密钥 SNS

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在资源上标有IAM用户的用户名时，您才能向IAM用户授予访问该资源的权限。有关更多信息，请参阅《IAM用户指南》中的IAM策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅《IAM用户指南》中的AWS 全局条件上下文密钥。

要查看亚马逊SNS条件密钥列表，请参阅《服务授权参考》中的 “亚马逊简单通知服务的条件密钥”。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Simple Notification Service 定义的资源。

要查看 Amazon SNS 基于身份的政策示例，请参阅。适用于 Amazon Simple Notification Service 的基于身份的策略示例

ACLs在亚马逊 SNS

支持ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACLs与基于资源的策略类似，尽管它们不使用JSON策略文档格式。

ABAC与亚马逊合作 SNS

支持ABAC（策略中的标签）：部分

基于属性的访问控制 (ABAC) 是一种基于属性定义权限的授权策略。在中 AWS，这些属性称为标签。您可以为IAM实体（用户或角色）和许多 AWS 资源附加标签。为实体和资源添加标签是的第一步。ABAC然后，您可以设计ABAC策略，允许在委托人的标签与他们尝试访问的资源上的标签匹配时进行操作。

ABAC在快速增长的环境中很有用，也有助于解决策略管理变得繁琐的情况。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关的更多信息ABAC，请参阅什么是ABAC？ 在《IAM用户指南》中。要查看包含设置步骤的教程ABAC，请参阅IAM用户指南中的使用基于属性的访问控制 (ABAC)。

在 Amazon 上使用临时证书 SNS

支持临时凭证：是

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关其他信息，包括哪些 AWS 服务 适用于临时证书 AWS 服务 ，请参阅《IAM用户指南》IAM中的 “适用于临时证书”。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM用户指南》中的切换到角色（控制台）。

您可以使用 AWS CLI 或手动创建临时证书 AWS API。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅中的临时安全证书IAM。

Amazon 的跨服务委托人权限 SNS

支持转发访问会话 (FAS)：是

当您使用IAM用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用调用委托人的权限 AWS 服务以及 AWS 服务 向下游服务发出请求的请求。FAS只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出请求。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。

Amazon 的服务角色 SNS

支持服务角色：是

服务IAM角色是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅《IAM用户指南》 AWS 服务中的创建角色以向委派权限。

> **Warning**
>
> 更改服务角色的权限可能会中断 Amazon 的SNS功能。仅当 Amazon SNS 提供相关指导时，才可编辑服务角色。

Amazon 的服务相关角色 SNS

支持服务相关角色：否

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅与之配合IAM使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

适用于 Amazon Simple Notification Service 的基于身份的策略示例

默认情况下，用户和角色无权创建或修改 Amazon SNS 资源。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或来执行任务 AWS API。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入角色。

要了解如何使用这些示例策略文档创建IAM基于身份的JSON策略，请参阅IAM用户指南中的创建IAM策略。

有关亚马逊SNS定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅服务授权参考中的亚马逊简单通知服务的操作、资源和条件密钥。ARNs

主题

策略最佳实践

基于身份的策略决定了是否有人可以在您的账户中创建、访问或删除亚马逊SNS资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM用户指南》中的AWS 托AWS 管策略或工作职能托管策略。
- 应用最低权限权限-使用IAM策略设置权限时，仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用应用权限IAM的更多信息，请参阅《IAM用户指南》IAM中的策略和权限。
- 使用IAM策略中的条件进一步限制访问权限-您可以在策略中添加条件以限制对操作和资源的访问权限。例如，您可以编写一个策略条件来指定所有请求都必须使用发送SSL。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM用户指南》中的IAMJSON策略元素：条件。
- 使用 A IAM ccess Analyzer 验证您的IAM策略以确保权限的安全性和功能性 — A IAM ccess Analyzer 会验证新的和现有的策略，以便策略符合IAM策略语言 (JSON) 和IAM最佳实践。IAMAccess Analyzer 提供了 100 多项策略检查和可行的建议，可帮助您制定安全和实用的策略。有关更多信息，请参阅《IAM用户指南》中的 IAMAccess Analyzer 策略验证。
- 需要多重身份验证 (MFA)-如果您的场景需要IAM用户或 root 用户 AWS 账户，请打开MFA以提高安全性。要要求MFA何时调用API操作，请在策略中添加MFA条件。有关更多信息，请参阅《IAM用户指南》中的配置MFA受保护的API访问权限。

有关中最佳做法的更多信息IAM，请参阅《IAM用户指南》IAM中的安全最佳实践。

使用亚马逊SNS控制台

要访问 Amazon Simple Notification Service 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 Amazon SNS 资源的详细信息 AWS 账户。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

您无需为仅拨打 AWS CLI 或的用户设置最低控制台权限 AWS API。相反，只允许访问与他们尝试执行的API操作相匹配的操作。

为确保用户和角色仍然可以使用亚马逊SNS控制台，还需要将亚马逊SNS*ConsoleAccess*或*ReadOnly* AWS 托管策略附加到实体。有关更多信息，请参阅《用户指南》中的向IAM用户添加权限。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界-权限边界是一项高级功能，您可以在其中设置基于身份的策略可以向IAM实体（IAM用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM用户指南》中的IAM实体的权限边界。
- 服务控制策略 (SCPs)-SCPs 是为中的组织或组织单位 (OU) 指定最大权限的JSON策略 AWS Organizations。 AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。SCP限制了成员账户中实体的权限，包括每个 AWS 账户 root 用户。有关 Organization SCPs s 和的更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅IAM用户指南中的策略评估逻辑。

允许用户查看他们自己的权限

此示例说明如何创建允许IAM用户查看附加到其用户身份的内联和托管策略的策略。此策略包括在控制台上或使用或以编程方式完成此操作的 AWS CLI 权限。 AWS API

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

亚马逊基于身份的政策 SNS

支持基于身份的策略：是

基于身份的策略是可以附加到身份（例如IAM用户、用户组或角色）的JSON权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅IAM用户指南中的创建IAM策略。

使用IAM基于身份的策略，您可以指定允许或拒绝的操作和资源，以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可以在JSON策略中使用的所有元素，请参阅IAM用户指南中的IAMJSON策略元素参考。

Amazon 基于身份的政策示例 SNS

要查看 Amazon SNS 基于身份的政策示例，请参阅。适用于 Amazon Simple Notification Service 的基于身份的策略示例

Amazon 内部基于资源的政策 SNS

支持基于资源的策略	是

基于资源的JSON策略是您附加到资源的策略文档。基于资源的策略的示例包括IAM角色信任策略和Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问权限，您可以将整个账户或另一个账户中的IAM实体指定为基于资源的策略中的委托人。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 AWS 账户，可信账户中的IAM管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM用户指南》IAM中的跨账户资源访问权限。

在 Amazon 上使用基于身份的政策 SNS

主题

- IAM和 Amazon SNS 政策合而为一
- Amazon SNS 资源ARN格式

- 亚马逊的SNSAPI行动
- 亚马逊SNS政策密钥
- 亚马逊政策示例 SNS

Amazon Simple Notification Service 与 AWS Identity and Access Management (IAM) 集成，因此您可以指定您的用户 AWS 账户 可以使用亚马逊SNS资源执行哪些亚马逊SNS操作。您可以指定策略中的特定主题。例如，您可以在创建策略时使用变量，该IAM策略授予组织中的某些用户对您的特定主题使用Publish操作的权限 AWS 账户。有关更多信息，请参阅《使用IAM》指南中的策略变量。

> **Important**
>
> 将亚马逊SNS与之搭配使用IAM并不会改变您使用亚马逊的方式SNS。亚马逊SNS操作没有变化，也没有与用户和访问控制相关的新亚马逊SNS操作。

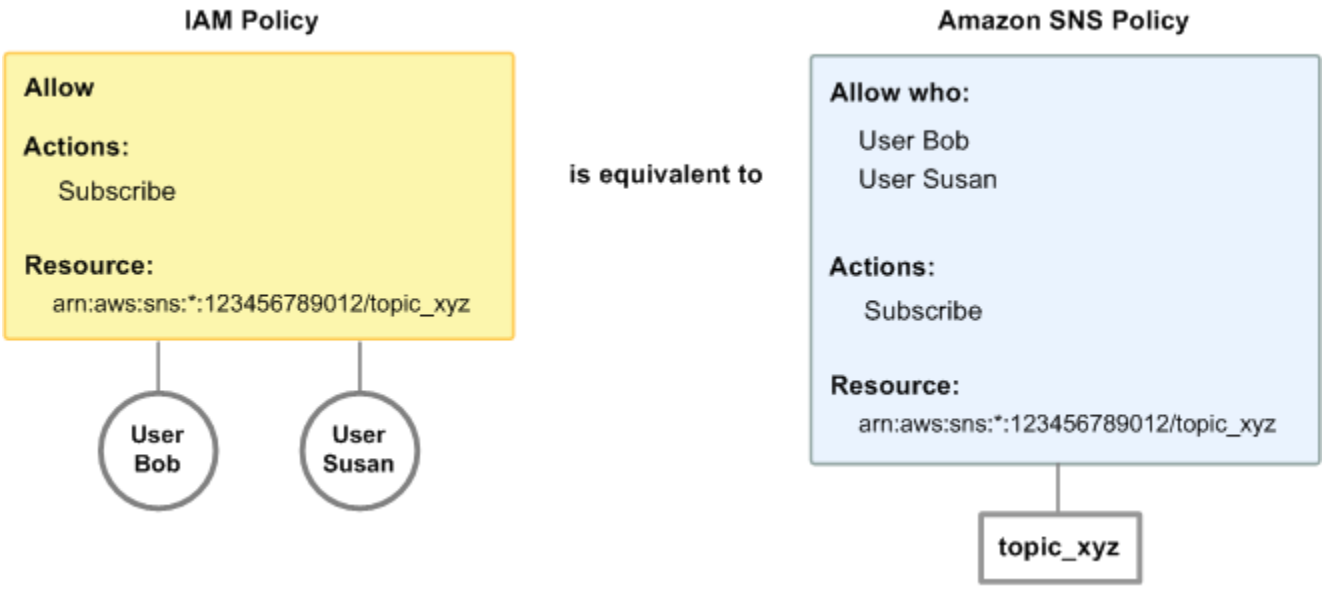
有关涵盖 Amazon SNS 操作和资源的政策示例，请参阅亚马逊政策示例 SNS。

IAM和 Amazon SNS 政策合而为一

您使用IAM政策来限制用户访问Amazon SNS 操作和主题。IAM策略只能限制您 AWS 账户内的用户访问权限，而不能限制其他用户的访问权限 AWS 账户。

您可以使用针对特定主题的 Amazon SNS 政策来限制谁可以处理该主题（例如，谁可以向该主题发布消息、谁可以订阅该主题等）。Amazon SNS 政策可以向其他 AWS 账户人或您自己的用户授予访问权限 AWS 账户。

要向您的用户授予访问您的亚马逊SNS主题的权限，您可以使用IAM政策、亚马逊SNS政策或两者兼而有之。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政策和 Amazon SNS 政策。该IAM政策允许亚马逊针对您的账户中名为 topic_xyz 的主题SNSSubscribe采取行动。 AWS 该IAM策略附加到用户 Bob 和 Susan（这意味着 Bob 和 Susan 拥有策略中规定的权限）。亚马逊SNS政策同样授予 Bob 和 Susan 访问 topic_x Subscribe yz 的权限。



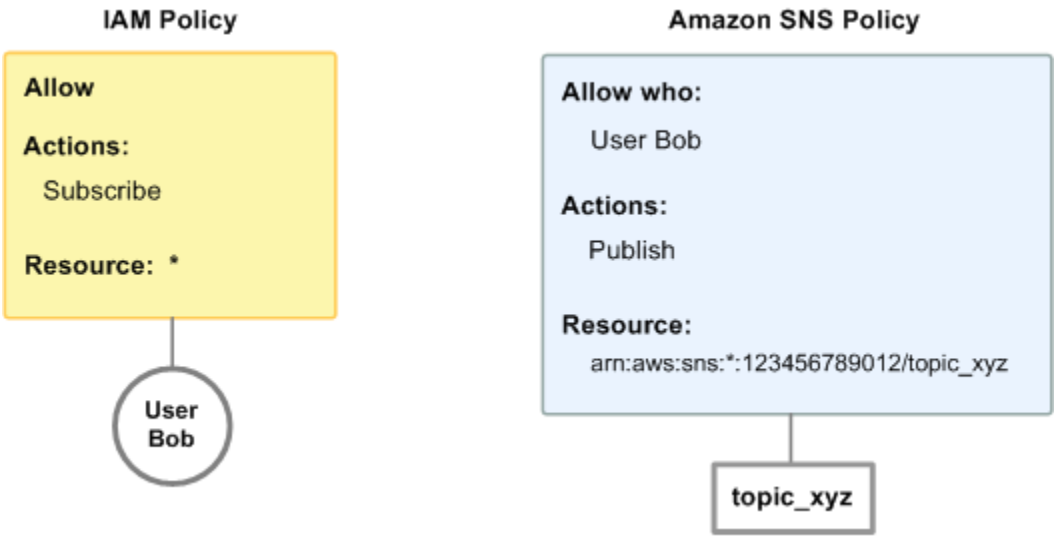
> **Note**
>
> 上述示例显示了不带任何条件的简单策略。您可以在上述任一策略中指定特定条件，并获得同样的结果。

AWS IAM和亚马逊SNS政策之间有一个区别：亚马逊SNS政策系统允许您向其他人授予权限 AWS 账户，而该IAM政策不允许您向其他人授予权限。

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。

Example 1

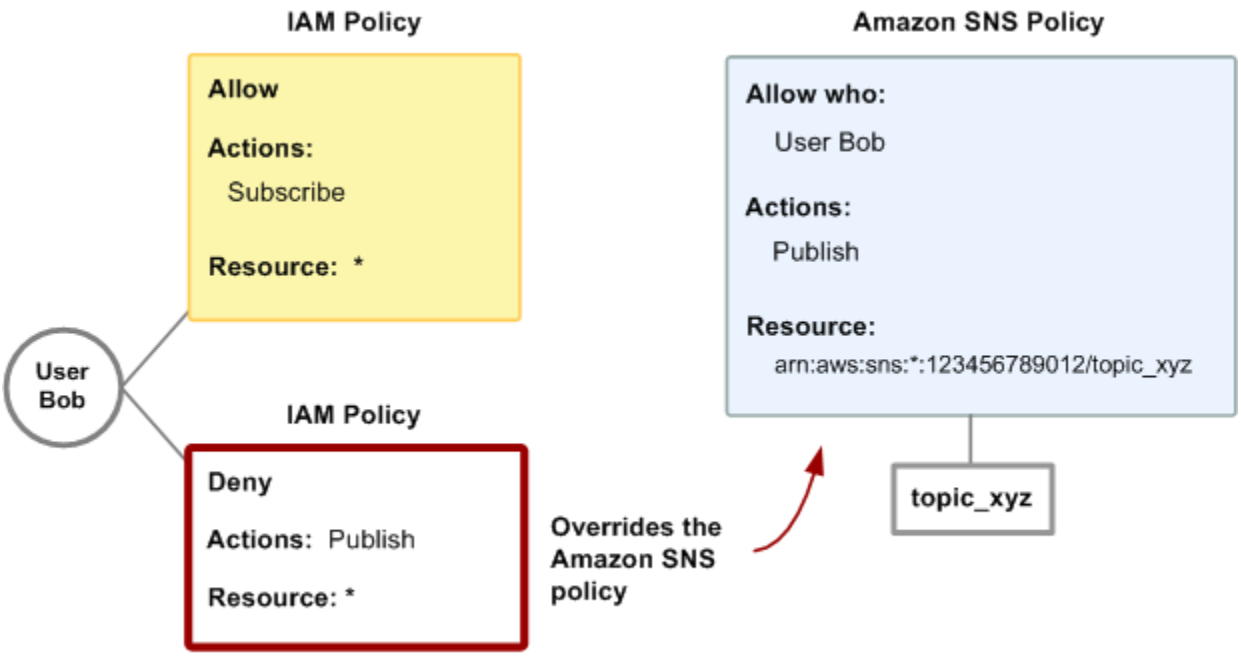
在此示例中，IAM政策和 Amazon SNS 政策都适用于 Bob。该IAM政策授予他`Subscribe`关于任何主题 AWS 账户的权限，而亚马逊SNS政策则授予他在特定主题`Publish`上使用的权限 (topic_xyz)。下图阐明了这一概念。



如果 Bob 发送订阅 AWS 账户中任何主题的请求，则IAM策略将允许该操作。如果 Bob 向 topic_xyz 发送发布消息的请求，亚马逊的SNS政策将允许该操作。

Example 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。让我们看看 Bob 无法向 topic_xyz 发布消息的情况，因此您要将其发布主题的功能全部删除。最简单的方法是添加一个IAM策略，拒绝他访问所有主题的`Publish`操作。第三项政策取代了最初允许他向 topic_xyz 发布内容的 Amazon SNS 政策，因为明确拒绝总是优先于允许（有关策略评估逻辑的更多信息，请参阅）。评估逻辑下图阐明了这一概念。



有关涵盖 Amazon SNS 操作和资源的政策示例，请参阅亚马逊政策示例 SNS。有关撰写亚马逊SNS政策的更多信息，请访问亚马逊技术文档SNS。

Amazon SNS 资源ARN格式

对于 AmazonSNS，主题是您可以在策略中指定的唯一资源类型。以下是主题的 Amazon 资源名称 (ARN) 格式。

```
arn:aws:sns:region:account_ID:topic_name
```

有关的更多信息ARNs，请转至ARNs《IAM用户指南》。

Example

以下是在 us-east-2 区域命名的 MyTopic 主题的，属ARN于 123456789012。 AWS 账户

```
arn:aws:sns:us-east-2:123456789012:MyTopic
```

Example

如果您 MyTopic 在 Amazon SNS 支持的每个不同地区都有一个名为的主题，则可以通过以下方式指定主题ARN。

```
arn:aws:sns:*:123456789012:MyTopic
```

您可以在主题名称中使用 * 和 ? 通配符。例如，以下可以参考由 Bob 创建、前缀为 bob_ 的所有主题。

```
arn:aws:sns:*:123456789012:bob_*
```

为方便起见，当您创建主题时，Amazon 会在响应ARN中SNS返回该主题的内容。

亚马逊的SNSAPI行动

在IAM政策中，您可以指定 Amazon SNS 提供的任何操作。但是，ConfirmSubscription和Unsubscribe操作不需要身份验证，这意味着即使您在策略中指定了这些操作，也IAM不会限制用户对这些操作的访问。

您在策略中指定的所有操作必须加上小写的字符串 sns: 为前缀。例如，要指定所有 Amazon SNS 操作，可以使用sns:*。要查看操作列表，请访问《Amazon 简单通知服务API参考》。

亚马逊SNS政策密钥

Amazon SNS 实施了以下 AWS 广泛的策略密钥，以及一些特定于服务的密钥。

有关每个条件键支持的条件键的列表 AWS 服务，请参阅《IAM用户指南》 AWS 服务中的操作、资源和条件键。有关可以用于多个条件键的列表 AWS 服务，请参阅《IAM用户指南》中的AWS 全局条件上下文密钥。

Amazon SNS 使用以下特定于服务的密钥。使用策略中限制访问 Subscribe 请求的这些密钥。

- sns: endpoint — URL、电子邮件地址，或者ARN来自Subscribe请求或之前确认的订阅。通过字符串条件（请参阅 亚马逊政策示例 SNS）限制访问特定终端节点（例如 *@yourcompany.com）。
- sns:protocol—来自 Subscribe 请求或之前已确认的订阅的 protocol 值。与字符串条件一起使用（请参阅 亚马逊政策示例 SNS），以限制向特定传输协议发布消息（例如，https）。

亚马逊政策示例 SNS

本节介绍几项用于控制用户访问Amazon的简单策略SNS。

> **Note**
>
> 将来，Amazon SNS 可能会根据该政策的既定目标，增加新的行动，这些措施在逻辑上应包含在以下政策中。

Example 1：允许群组创建和管理主题

在本示例中，我们创建授权访问 `CreateTopic`、`ListTopics`、`SetTopicAttributes` 和 `DeleteTopic` 的策略。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2：允许 IT 群组向特定主题发布消息

在本示例中，我们为 IT 创建群组，并将授权访问 `Publish` 的策略分配至相关特定主题上。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3：让用户 AWS 账户 能够订阅主题

在示例中，我们创建授予访问 `Subscribe` 操作的权限的策略，以及针对 `sns:Protocol` 和 `sns:Endpoint` 策略密钥的字符串匹配条件。

```
{
  "Statement": [{
    "Effect": "Allow",
```

```
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "SNS:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4：允许合作伙伴向特定主题发布消息

您可以使用 Amazon SNS 政策或IAM政策来允许合作伙伴针对特定主题发布内容。如果您的合作伙伴有 AWS 账户，则使用Amazon SNS 政策可能会更容易。但是，合作伙伴公司中任何拥有 AWS 安全证书的人都可以发布有关该主题的消息。本示例假设您想要对特定人员进行访问限制（或应用程序限制）。为此，您需要像对待自己公司的用户一样对待合作伙伴，并使用IAM政策而不是亚马逊SNS政策。

在此示例中，我们创建了一个名 WidgetCo 为代表合作伙伴公司的群组；我们为合作伙伴公司中需要访问权限的特定人员（或应用程序）创建一个用户；然后将该用户放入该群组。

然后，我们附加一个策略，授予该群组对名为的特定主题的Publish访问权限WidgetPartnerTopic。

我们还想阻止该 WidgetCo 小组对主题做任何其他事情，因此我们添加了一份声明，拒绝允许除Publish此之外的任何主题以外的任何亚马逊SNS操作除外 WidgetPartnerTopic。只有在系统其他地方有一项广泛政策，允许用户广泛访问亚马逊时，才需要这样做SNS。

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
```

```
}
```

在 Amazon 上使用临时安全证书 SNS

AWS Identity and Access Management (IAM) 允许您向需要访问您的 AWS 资源的用户和应用程序授予临时安全证书。这些临时安全证书主要用于通过行业标准协议（例如和 OpenID Conn OIDC ect ()）进行IAM角色SAML和联合访问。

要有效地管理对 AWS 资源的访问权限，必须了解以下关键概念：

- IAM角色-角色用于委派对 AWS 资源的访问权限。角色可以由 Amazon 实EC2例、Lambda 函数等实体或其他用户担任。 AWS 账户
- 联合用户-这些用户是通过外部身份提供商 (IdPs) 使用SAML或进行身份验证的用户OIDC。建议人类用户使用联合访问权限，而软件应用程序则应使用IAM角色。
- 无@@ 处不在的角色-对于需要 AWS 访问权限的外部应用程序，您可以使用 IAM Roles Anywhere 来安全地管理访问权限，而无需创建长期凭证。

您可以使用临时安全证书向 Amazon 提出请求SNS。SDKs和API库使用这些凭据计算必要的签名来验证您的请求。凭证过期的请求将被亚马逊拒绝SNS。

有关临时安全证书的更多信息，请参阅《用户指南》IAM中的使用角色和向经过外部身份验证的用户提供访问权限（身份联合IAM）。

Example HTTPS请求示例

以下示例演示如何使用从 AWS Security Token Service (STS) 中获取的临时安全证书对 Amazon SNS 请求进行身份验证。

```
https://sns.us-east-2.amazonaws.com/
?Action=CreateTopic
&Name=My-Topic
&SignatureVersion=4
&SignatureMethod=AWS4-HMAC-SHA256
&Timestamp=2023-07-05T12:00:00Z
&X-Amz-Security-Token=SecurityTokenValue
&X-Amz-Date=20230705T120000Z
&X-Amz-Credential=<your-access-key-id>/20230705/us-east-2/sns/aws4_request
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

对请求进行身份验证的步骤

1. 获取临时安全证书- AWS STS 用于代入角色或获取联合用户证书。这将为您提供访问密钥 ID、私有访问密钥和安全令牌。
2. 构造请求 — 包括您的 Amazon SNS 操作所需的参数（例如 CreateTopic），并确保您使用这些参数HTTPS进行安全通信。
3. 签署请求-使用 AWS 签名版本 4 流程签署您的请求。这包括创建规范请求 string-to-sign，然后计算签名。有关 AWS 签名版本 4 的更多信息，请参阅 Amazon EBS 用户指南中的使用签名版本 4 签名。
4. 发送请求 — X-Amz-Security-Token 在您的请求标题中加入，以便将临时安全证书传递给 Amazon SNS。

Amazon SNS API 权限：操作和资源参考

以下列表提供了特定于 Amazon SNS 实施访问控制的信息：

- 每个策略都必须仅覆盖单一主题（在编写一个策略时，请勿包括覆盖不同主题的语句）
- 每个策略必须有一个独立的策略`Id`
- 策略中的每个语句必须是一个独立的语句`sid`

策略配额

以下表格列举了策略语句的最大配额。

名称	最大配额
字节	30 kb
语句	100
委托人	1 到 200（0 无效。）
资源	1 (0) 无效。 该值必须与策略主题ARN的值相匹配。）

有效的亚马逊SNS政策行动

Amazon SNS 支持下表所示的操作。

操作	描述
sns: AddPermission	授予向主题策略添加权限的权限。
sns: DeleteTopic	授予删除一个主题的权限。
sns: GetDataProtectionPolicy	授予权限以检索主题的数据保护策略。
sns: GetTopicAttributes	授予获取所有主题属性的权限。
sns: ListSubscriptionsByTopic	授予检索对特定主题的所有订阅的权限。
sns: ListTagsForResource	授予列出添加到特定主题的所有标签的权限。
sns:Publish	授予向主题或终端节点发布和批量发布内容的权限。有关更多信息，请参阅《Amazon 简单通知服务API参考》PublishBatch中的“发布” 和。
sns: PutDataProtectionPolicy	授予权限以设置主题的数据保护策略。
sns: RemovePermission	授予在主题策略中取消任何权限的权限。
sns: SetTopicAttributes	授予设置一个主题的属性的权限。
sns:Subscribe	授予订阅一个主题的权限。

特定于服务的密钥

Amazon SNS 使用以下特定于服务的密钥。您可以在限制访问 `Subscribe` 请求的策略中使用它们。

- sns: endpoint — URL、电子邮件地址，或者ARN来自`Subscribe`请求或之前确认的订阅。与字符串条件一起使用（请参阅 亚马逊政策示例 SNS），以限制对某个特定终端节点的访问（例如，*@example.com）。
- sns:protocol—来自 `Subscribe` 请求或之前已确认的订阅的 `protocol` 值。与字符串条件一起使用（请参阅 亚马逊政策示例 SNS），以限制向特定传输协议发布消息（例如，https）。

> **Important**
>
> 当您使用策略控制 sns: Endpoint 的访问权限时，请注意DNS问题将来可能会影响终端节点的名称解析。

Amazon Simple Notification Service 身份和访问故障排查

使用以下信息来帮助您诊断和修复与Amazon合作时可能遇到的常见问题，SNS以及IAM。

主题

- 我无权在 Amazon 上执行任何操作 SNS
- 我无权执行 iam：PassRole
- 我想允许我以外的人访问我的 AWS 账户 Amazon SNS 资源

我无权在 Amazon 上执行任何操作 SNS

如果您收到一个错误，指明您无权执行某个操作，则必须更新策略以允许您执行该操作。

当 `mateojackson` 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `sns:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

在此情况下，Mateo 的策略必须更新以允许其使用 `sns:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果您收到错误消息，说您无权执行该`iam:PassRole`操作，则必须更新您的政策，以允许您将角色传递给亚马逊SNS。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的IAM用户`marymajor`尝试使用控制台在 Amazon 中执行操作时，会出现以下示例错误SNS。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人访问我的 AWS 账户 Amazon SNS 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon 是否SNS支持这些功能，请参阅亚马逊是如何SNS与之合作的 IAM。
- 要了解如何提供对您拥有的资源的访问权限，请参阅《IAM用户指南》中的 AWS 账户 向其他IAM用户提供访问权限。 AWS 账户
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅IAM用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过联合身份验证提供访问权限，请参阅《用户指南》中的向经过外部身份验证的用户提供访问权限（联合身份验证）。IAM
- 要了解使用角色和基于资源的策略进行跨账户访问的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。

在 Amazon 中记录和监控 SNS

本主题提供有关如何记录和监控 Amazon SNS 主题的说明，以保持对消息基础设施的可见性和控制力。首先，您将了解如何使用记录Amazon SNS API 通话 AWS CloudTrail。此日志允许您跟踪对您的 Amazon SNS 主题执行的操作，例如主题创建、订阅管理和消息发布。通过分析 CloudTrail 日志，您可以确定谁提出了特定API请求以及这些请求是何时提出的，从而帮助您审计和排查您的 Amazon SNS 使用情况。

接下来，您将探索如何使用亚马逊监控亚马逊SNS话题 CloudWatch。 CloudWatch 提供的指标可让您实时观察您的 Amazon SNS 主题的表现和运行状况。您将学习如何根据这些指标设置警报，从而能够对任何异常情况做出及时响应，例如传送失败或消息延迟过长。这种监控功能可确保您可以通过主动解决潜在问题来保持SNS基于您的消息系统的可靠性。

主题

使用记录亚马逊SNSAPI通话 CloudTrail

SNSAmazon 与 AWS CloudTrail一项服务集成，该服务可记录用户、角色或 AWS 服务在亚马逊中执行的操作SNS。 CloudTrail 将 Amazon 的API呼叫记录SNS为事件。捕获的调用包括来自亚马逊SNS控制台的调用和对亚马逊SNSAPI运营的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail事件持续传输到 Amazon S3 存储桶，包括针对亚马逊的事件SNS。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。通过收集的信息 CloudTrail，您可以确定向亚马逊发出的请求SNS、发出请求的 IP 地址、谁提出请求、何时提出请求以及其他详细信息。

要了解更多信息 CloudTrail，包括如何配置和启用它，请参阅《AWS CloudTrail 用户指南》。

亚马逊SNS信息位于 CloudTrail

CloudTrail 在您创建账户 AWS 账户 时已在您的账户上启用。当 Amazon 中出现支持的事件活动时SNS，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在中查看、搜索和下载最近发生的事件 AWS 账户。有关更多信息，请参阅使用事件历史记录查看 CloudTrail事件。

要持续记录您的 AWS 账户事件（包括 Amazon 的事件）SNS，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，当您在控制台中创建跟踪时，该跟踪将应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析 CloudTrail 日志中收集的事件数据并对其采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 为以下各项配置亚马逊SNS通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

在中控制飞机事件 CloudTrail

Amazon SNS 支持将以下操作作为事件记录在 CloudTrail 日志文件中：

- `AddPermission`

- CheckIfPhoneNumberIsOptedOut
- ConfirmSubscription
- CreatePlatformApplication
- CreatePlatformEndpoint
- CreateSMSSandboxPhoneNumber
- CreateTopic
- DeleteEndpoint
- DeletePlatformApplication
- DeleteSMSSandboxPhoneNumber
- DeleteTopic
- GetDataProtectionPolicy
- GetEndpointAttributes
- GetPlatformApplicationAttributes
- GetSMSAttributes
- GetSMSSandboxAccountStatus
- GetSubscriptionAttributes
- GetTopicAttributes
- ListEndpointsByPlatformApplication
- ListOriginationNumbers
- ListPhoneNumbersOptedOut
- ListPlatformApplications
- ListSMSSandboxPhoneNumbers
- ListSubscriptions
- ListSubscriptionsByTopic
- ListTagsForResource
- ListTopics
- OptInPhoneNumber
- PutDataProtectionPolicy
- RemovePermission
- SetEndpointAttributes

- SetPlatformApplicationAttributes
- SetSMSAttributes
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource
- VerifySMSSandboxPhoneNumber

> **Note**
>
> 如果您未登录到 Amazon Web Services（未经身份验证的模式），并且调用了ConfirmSubscription或取消订阅操作，则这些操作将不会被登录到。 CloudTrail例如，当您选择电子邮件通知中提供的链接确认对某一主题的待确认订阅时，会在未验证模式下调用 ConfirmSubscription 操作。在此示例中，ConfirmSubscription操作不会被记录到 CloudTrail。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用 root 还是 AWS Identity and Access Management (IAM) 用户凭据发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅CloudTrail userIdentity 元素。

中的数据平面事件 CloudTrail

要启用 CloudTrail 文件中以下API操作的日志记录，您需要在中启用数据平面API活动的日志记录 CloudTrail。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的记录数据事件。

还可以按资源类型筛选数据平面事件，以便精细控制您要选择性登录和付费的 Amazon SNS API 调用哪个 Amazon 电话。 CloudTrail例如，通过指定AWS::SNS::Topic为资源类型，您可以记录对主题的调用Publish和PublishBatchAPI操作。同样，通过指定AWS::SNS::PlatformEndpoint为

资源类型，您可以记录对平台终端节点的 Publish API 操作的调用。有关更多信息，请参阅 “ AWS CloudTrail API参考” AdvancedEventSelector中的。

> **Note**
>
> Amazon SNS 资源类型AWS::SNS::PhoneNumber不是由记录的 CloudTrail。

亚马逊SNS数据平面 APIs

- Publish
- PublishBatch

示例：Amazon SNS 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。 CloudTrail 日志文件不是公共API调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了一个演示ListTopicsCreateTopic、和DeleteTopic操作的 CloudTrail 日志条目。


```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "ListTopics",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
```

```
      "requestParameters": {
        "nextToken": "ABCDEF1234567890EXAMPLE=="
      },
      "responseElements": null,
      "requestID": "example1-b9bb-50fa-abdb-80f274981d60",
      "eventID": "example0-09a3-47d6-a810-c5f9fd2534fe",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "CreateTopic",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "name": "hello"
      },
      "responseElements": {
        "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
      },
      "requestID": "example7-5cd3-5323-8a00-f1889011fee9",
      "eventID": "examplec-4f2f-4625-8378-130ac89660b1",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
```

```
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "DeleteTopic",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
      },
      "responseElements": null,
      "requestID": "example5-4faa-51d5-aab2-803a8294388d",
      "eventID": "example8-6443-4b4d-abfd-1b867280d964",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
  ]
}
```

以下示例显示了演示Publish和PublishBatch操作的 CloudTrail 日志条目。

Publish

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "ExampleUser"
   },
   "attributes": {
    "creationDate": "2023-08-21T16:44:05Z",
    "mfaAuthenticated": "false"
   }
```

```
  }
 },
 "eventTime": "2023-08-21T16:48:37Z",
 "eventSource": "sns.amazonaws.com",
 "eventName": "Publish",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
 "requestParameters": {
  "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
  "message": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "subject": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "messageStructure": "json",
  "messageAttributes": "HIDDEN_DUE_TO_SECURITY_REASONS"
 },
 "responseElements": {
  "messageId": "0787cd1e-d92b-521c-a8b4-90434e8ef840"
 },
 "requestID": "0a8ab208-11bf-5e01-bd2d-ef55861b545d",
 "eventID": "bb3496d4-5252-4660-9c28-3c6aebdb21c0",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```

PublishBatch

```
{
 "eventVersion": "1.09",
```

```
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "ExampleUser"
   },
   "attributes": {
    "creationDate": "2023-08-21T19:20:49Z",
    "mfaAuthenticated": "false"
   }
  }
 },
 "eventTime": "2023-08-21T19:22:01Z",
 "eventSource": "sns.amazonaws.com",
 "eventName": "PublishBatch",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
 "requestParameters": {
  "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
  "publishBatchRequestEntries": [{
    "id": "1",
    "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
   },
   {
    "id": "2",
    "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
   }
  ]
 },
 "responseElements": {
  "successful": [{
    "id": "1",
    "messageId": "30d68101-a64a-5573-9e10-dc5c1dd3af2f"
```

```
    },
    {
     "id": "2",
     "messageId": "c0aa0c5c-561d-5455-b6c4-5101ed84de09"
    }
   ],
   "failed": []
  },
  "requestID": "e2cdf7f3-1b35-58ad-ac9e-aaaea0ace2f1",
  "eventID": "10da9a14-0154-4ab6-b3a5-1825b229a7ed",
  "readOnly": false,
  "resources": [{
   "accountId": "123456789012",
   "type": "AWS::SNS::Topic",
   "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
  }],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
   "tlsVersion": "TLSv1.2",
   "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
   "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
  }
 }
```

使用监控亚马逊SNS话题 CloudWatch

亚马逊SNS和亚马逊 CloudWatch 是集成在一起的，因此您可以收集、查看和分析每条有效的亚马逊SNS通知的指标。 CloudWatch 为亚马逊配置完毕后SNS，您可以更好地了解亚马逊SNS主题、推送通知和SMS交付的表现。例如，您可以设置警报，以便在 Amazon SNS 指标达到指定阈值时向您发送电子邮件通知，例如NumberOfNotificationsFailed。有关 Amazon SNS 发送到的所有指标的列表 CloudWatch，请参阅亚马逊SNS指标。有关 Amazon SNS 推送通知的更多信息，请参阅通过 Amazon 发送移动推送通知 SNS。

> **Note**
>
> 系统会自动收集您 CloudWatch 为 Amazon SNS 主题配置的指标，并每隔 1 分钟推送 CloudWatch 一次。这些指标是针对所有符合活跃 CloudWatch 指南的话题收集的。自该主题 CloudWatch 的上次活动（即任何API通话）起，该话题最多六小时被视为处于活动状态。

中报告的亚马逊SNS指标不收取任何费用 CloudWatch；它们是作为亚马逊SNS服务的一部分提供的。

查看 Amazon 的 CloudWatch 指标 SNS

您可以使用 CloudWatch 控制台、 CloudWatch自己的命令行界面 (CLI) 或以编程方式SNS使用来监控 Amazon 的 CloudWatch API指标。以下过程展示如何使用 AWS Management Console访问指标。

使用 CloudWatch 控制台查看指标

1. 登录CloudWatch控制台。
2. 在导航面板上，选择 Metrics。
3. 在所有指标选项卡上 SNS，选择，然后选择以下维度之一：
   - 国家、SMS类型
   - PhoneNumber
   - Topic Metrics (主题指标)
   - Metrics with no dimensions (无维度指标)
4. 要查看详细信息，请选择特定项目。例如，如果您选择 “主题指标”，然后选择 NumberOfMessagesPublished，则会显示在 6 小时的时间范围内 1 分钟内发布的 Amazon SNS 消息的平均数量。
5. 要查看亚马逊SNS使用量指标，请在所有指标选项卡上，选择使用情况，然后选择目标亚马逊SNS使用量指标（例如`NumberOfMessagesPublishedPerAccount`）。

为 Amazon SNS 指标设置 CloudWatch 警报

CloudWatch 还允许您在指标达到阈值时设置警报。例如，您可以为指标设置警报 NumberOfNotificationsFailed，以便在采样周期内达到指定的阈值时，系统会发送一封电子邮件通知来通知您该事件。

使用 CloudWatch 控制台设置警报

1. 登录 AWS Management Console 并打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 选择警报，然后选择创建警报按钮。这样会启动“Create Alarm”向导。
3. 滚动浏览 Amazon SNS 指标，找到您要设置警报的指标。选择该指标创建一个警报并选择继续。

4. 填写指标的名称、描述、阈值、时间值，然后选择继续。
5. 按照报警器说明选择“Alarm”。如果您 CloudWatch 想在达到警报状态时向您发送电子邮件，请选择现有的 Amazon SNS 主题或选择 “创建新电子邮件主题”。如果您选择 新建电子邮件主题，则可以为新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。选择继续。

> **Note**
>
> 如果您使用 “创建新电子邮件主题” 创建新的 Amazon SNS 主题，则必须先验证电子邮件地址，然后他们才能收到通知。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，“Create Alarm”向导会给您一次机会检查您准备创建的报警器。如果你想做一些变动，那么您可使用右边的“Edit”链接。如果您满意，请选择创建警报。

有关使用 CloudWatch 和警报的更多信息，请参阅CloudWatch文档。

亚马逊SNS指标

Amazon SNS 将以下指标发送至 CloudWatch。

命名空间	指标	描述
AWS/SNS	NumberOfMessagesPublished	发布到您的 Amazon SNS 主题的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和
AWS/SNS	NumberOfNotificationsDelivered	从您的 Amazon SNS 主题成功发送到订阅终端节点的消息数量。 要想成功传输，终端节点的订阅必须接受消息。在以下两种情况下订阅可接受消息：a.) 它缺少筛选策略或 b.)

命名空间	指标	描述
		其筛选策略中包含的属性与分配给消息的属性相匹配。如果订阅拒绝消息，则传输尝试不会计入此指标。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和
AWS/SNS	`NumberOfNotificationsFailed`	Amazon SNS 未能传送的消息数量。 对于亚马逊SQSSMS、电子邮件或移动推送终端节点，当亚马逊SNS停止尝试发送消息时，该指标将以 1 为增量。对于我们的HTTPHTTPS终端节点，该指标包括每一次失败的交付尝试，包括初次尝试之后的重试次数。对于所有其他终端节点，消息传输失败时计数增加 1（不考虑尝试次数）。 此指标不包括被订阅筛选策略拒绝的消息。 您可以控制HTTP终端节点的重试次数。有关更多信息，请参阅 Amazon SNS 消息传送重试次数。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和、平均值

命名空间	指标	描述
AWS/SNS	NumberOfNotificationsFilteredOut	被订阅筛选策略拒绝的消息数量。如果消息属性与策略属性不匹配，筛选策略会拒绝消息。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和、平均值
AWS/SNS	NumberOfNotificationsFilteredOut-MessageAttributes	被基于属性的筛选的订阅筛选策略拒绝的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和、平均值
AWS/SNS	NumberOfNotificationsFilteredOut-MessageBody	被基于有效负载的筛选的订阅筛选策略拒绝的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和、平均值

命名空间	指标	描述
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidAttributes	由于消息的属性无效（例如，因为属性格式不正确）而被订阅筛选策略拒绝的邮件数量。JSON 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和、平均值
AWS/SNS	NumberOfNotificationsFilteredOut-NoMessageAttributes	由于消息没有属性而被订阅筛选策略拒绝的消息数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和、平均值
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidMessageBody	由于邮件正文无法进行过滤（例如，无效的邮件正文）而被订阅筛选策略拒绝的JSON邮件数量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和、平均值

命名空间	指标	描述
AWS/SNS	NumberOfNotificationsRedrivenToDlq	已移动到死信队列的消息量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和、平均值
AWS/SNS	NumberOfNotificationsFailedToRedriveToDlq	无法移动到死信队列中的消息量。 单位：计数 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：总和、平均值
AWS/SNS	PublishSize	已发布消息的大小。 单位：字节 有效维度：应用程序 PhoneNumber、平台和 TopicName 有效统计数据：最小值、最大值、平均值和计数

命名空间	指标	描述
AWS/SNS	SMSMonthToDateSpentUSD	自当前日历月开始以来您因发送SMS消息而产生的费用。 您可以为该指标设置警报，以了解您的 month-to-date 费用何时接近您账户的每月SMS支出配额。当 Amazon SNS 确定发送SMS消息的费用超过此配额时，它会在几分钟内停止发布SMS消息。 有关设置每月SMS支出配额的信息，或有关申请增加支出配额的信息 AWS，请参阅在 Amazon 中设置SMS消息偏好 SNS。 单位：USD 有效尺寸：无 有效统计数据：总和
AWS/SNS	SMSSuccessRate	成功SMS传送消息的比率。 单位：计数 有效尺寸： PhoneNumber 有效统计数据：总和、平均值、数据样本

Amazon SNS 指标的尺寸

Amazon 简单通知服务会将以下维度发送至 CloudWatch。

维度	描述
Application	对应用程序对象进行筛选，这些对象表示在支持的推送通知服务之一中注册的应用程序和设备，例如APNs和FCM。
Application,Platform	对应用程序和平台对象进行筛选，其中平台对象用于支持的推送通知服务，例如APNs和FCM。
Country	按SMS邮件的目的地国家或地区进行筛选。国家或地区由其 ISO 3166-1 alpha-2 代码表示。
PhoneNumber	当您SMS直接向电话号码（不含主题）发布内容时，会根据电话号码进行筛选。
Platform	筛选推送通知服务的平台对象，例如APNs和FCM。
TopicName	筛选亚马逊SNS主题名称。
SMSType	筛选消息的SMS消息类型。可以为 promotional 或 transactional。

Amazon SNS 使用量指标

Amazon 简单通知服务向发送以下使用量指标 CloudWatch。

命名空间	服务	指标	资源	类型	描述
AWS/Usage	SNS	ResourceCount	NumberOfMessagesPublishedPerAccount	资源	• 在您的 AWS 账户中发布到您的 Amazon SNS 主题的消息数量。 • 单位：无 • 有效统计数据：Sum

命名空间	服务	指标	资源	类型	描述
AWS/Usage	SNS	ResourceCount	ApproximateNumberOfTopics	资源	• 您 AWS 账号中话题的大致数量。 • 单位：无 • 有效统计数据：Average、Minimum、Maximum、Sum
AWS/Usage	SNS	ResourceCount	ApproximateNumberOfFilterPolicies	资源	• 您的 AWS 账户中的大致筛选条件策略数量。 • 单位：无 • 有效统计数据：Average、Minimum、Maximum、Sum
AWS/Usage	SNS	ResourceCount	ApproximateNumberOfPendingSubscriptions	资源	• 您 AWS 账户中待处理订阅的大致数量。 • 单位：无 • 有效统计数据：Average、Minimum、Maximum、Sum

命名空间	服务	指标	资源	类型	描述
AWS/Usage	SNS	CallCount	• AddPermission • CheckIfPhoneNumberIsOptedOut • CreatePlatformApplication • CreatePlatformEndpoint • ConfirmSubscription • CreateSMSSandboxPhoneNumber • CreateTopic • DeleteEndpoint • DeletePlatformApplication • DeleteSMSSandboxPhoneNumber • DeleteTopic	API	• 您 AWS 账户中选定的 Amazon SNS API 的API通话次数。 • 单位：无 • 有效统计数据：Sum

命名空间	服务	指标	资源	类型	描述
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

命名空间	服务	指标	资源	类型	描述
			• ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes • SetSubscriptionAttributes • SetTopicAttributes		

命名空间	服务	指标	资源	类型	描述
			• Subscribe • Unsubscribe • UntagResource • VerifySMSSandboxPhoneNumber		

Amazon 合规性验证 SNS

第三方审计师评估亚马逊SNS的安全与合规性，这是多项 AWS 合规计划的一部分，包括《健康保险流通与责任法案》(HIPAA)。

有关特定合规计划范围内的 AWS 服务列表，请参阅按合规计划划分的范围内的AWSAWS 服务按合规计划。有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 Amazon SNS 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在上部署以安全性和合规性为重点的基准环境的步骤。 AWS
- HIPAA安全与合规架构白皮书 — 本白皮书描述了公司如何使用 AWS 来创建HIPAA符合要求的应用程序。
- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 此 AWS 服务可全面了解您的安全状态 AWS ，帮助您检查是否符合安全行业标准和最佳实践。

Amazon 的弹性 SNS

通过利用 AWS 全球基础设施（围绕可用区）来确保 Amazon SNS 的弹 AWS 区域 性。 AWS 区域 提供物理隔离和隔离的可用区，通过低延迟、高吞吐量和高度冗余的网络连接。这种架构允许在可用区之间进行无缝故障转移，而不会中断，与传统的数据中心基础架构相比，应用程序和数据库本质上更具容错性和可扩展性。通过使用可用区，Amazon SNS 订阅者可以从增强的可用性和可靠性中受益，即使存在潜在中断，也能保证消息传送。有关 AWS 区域 和可用区的更多信息，请参阅AWS 全球基础设施。

此外，Amazon SNS 主题的订阅可以配置传送重试和死信队列，从而自动处理暂时性故障，并确保消息可靠地到达其预期目的地。

Amazon SNS 还支持消息筛选和消息属性，这使您可以根据其特定用例定制弹性策略，从而增强应用程序的整体稳定性。

Amazon 的基础设施安全 SNS

作为一项托管服务，SNSAmazon 受《安全、身份与合规最佳实践》文档中的 AWS 全球网络安全程序的保护。

使用 AWS API操作SNS通过网络访问 Amazon。客户端必须支持传输层安全 (TLS) 1.2 或更高版本。客户端还必须支持带有 Perfect Forward Secrecy (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman () 或 Elliptic Curve Ephemeral Diffie-Hellman ()。DHE ECDHE

您必须同时使用访问密钥 ID 和与IAM委托人关联的私有访问密钥签署请求。或者，您可以使用 AWS Security Token Service (AWS STS) 生成用于签名请求的临时安全凭证。

您可以从任何网络位置调用这些API操作，但是 Amazon SNS 支持基于资源的访问策略，其中可能包括基于源 IP 地址的限制。您还可以使用亚马逊SNS策略来控制来自特定亚马逊VPC终端节点或特定终端节点的访问VPCs。这有效地将对给定 Amazon SNS 主题的网络访问与 AWS 网络VPC中的特定主题隔离开来。有关更多信息，请参阅 限制仅从特定VPC终端节点发布到 Amazon SNS 主题。

Amazon SNS 安全最佳实践

AWS 为 Amazon 提供了许多安全功能SNS。在您自己的安全策略的上下文中查看这些安全功能。

> **Note**
>
> 有关这些安全功能的指南适用于常见的应用场景和实施。我们建议您在特定应用场景、架构和威胁模型的上下文中查看这些最佳实践。

预防性最佳实践

以下是 Amazon SNS 的预防性安全最佳实践。

主题

确保主题不可公开访问

除非您明确要求互联网上的任何人能够阅读或写入您的Amazon SNS 主题，否则应确保您的主题不可公开访问（世界上所有人或任何经过身份验证的 AWS 用户都可以访问）。

- 避免创建 `Principal` 设置为 `""` 的策略。
- 避免使用通配符 (*)。相反，对一个特定用户或多个用户命名。

实施最低权限访问

授予权限时，您可以决定谁获得权限、权限适用于哪些主题以及您希望允许对这些主题执行的特定API操作。实施最低权限原则对于降低安全风险至关重要。它还有助于减少错误或恶意意图的负面影响。

遵循授予最低权限的标准安全建议。也就是说，只授予执行特定任务所需的权限。您可以通过使用与用户访问相关的安全策略组合来实施最低权限。

Amazon SNS 使用出版商-订阅模式，需要三种类型的用户账户访问权限：

- 管理员 – 用于创建、修改和删除主题的访问权。管理员还控制主题策略。
- 发布者 – 用于向主题发送消息的访问权。
- 订阅者 – 用于订阅主题的访问权。

有关详细信息，请参阅以下章节：

- Amazon 中的身份和访问管理 SNS
- Amazon SNS API 权限：操作和资源参考

为需要 Amazon SNS 访问权限的应用程序和 AWS 服务使用IAM角色

要使应用程序或 AWS 服务（例如亚马逊EC2）访问亚马逊SNS主题，它们必须在 AWS API请求中使用有效的 AWS 凭证。由于这些证书不会自动轮换，因此您不应将 AWS 凭证直接存储在应用程序或EC2实例中。

您应该使用IAM角色来管理需要访问 Amazon 的应用程序或服务的临时证书SNS。使用角色时，您无需向EC2实例或 AWS 服务分配长期证书（例如用户名、密码和访问密钥），例如 AWS Lambda。相反，该角色提供临时权限，供应用程序在调用其他 AWS 资源时使用。

有关更多信息，请参阅IAM《用户指南》中的角色和角色的常见场景：用户、应用程序和服务。IAM

实施服务器端加密

要减少数据泄漏问题，可以通过静态加密，使用存储在与消息存储位置不同的位置的密钥来对消息进行加密。服务器端加密 (SSE) 提供静态数据加密。Amazon 在存储您的数据时会在消息级别对其SNS进行加密，并在您访问消息时为您解密消息。SSE使用中管理的密钥 AWS Key Management Service。当您对请求进行身份验证并具有访问权限时，访问加密主题和未加密主题之间没有区别。

有关更多信息，请参阅 使用服务器端加密保护 Amazon SNS 数据 和 管理 Amazon SNS 加密密钥和成本。

实施传输中数据加密

可以使用发布传输过程中未加密的消息，但不建议这样做HTTP。但是，当使用对主题进行静态加密时AWS KMS，必须使用HTTPS该主题来发布消息，以确保静态和传输过程中的加密。虽然该主题不会自动拒绝HTTP消息，HTTPS但必须使用来维护安全标准。

AWS 建议你HTTPS改用HTTP。使用时HTTPS，即使SNS主题本身未加密，消息也会在传输过程中自动加密。否则HTTPS，基于网络的攻击者可以窃听网络流量或使用诸如之类的攻击来操纵网络流量。man-in-the-middle

要仅强制使用加密连接HTTPS，请在附加到未加密SNS主题的IAM策略中添加aws:SecureTransport条件。这会迫使消息发布者HTTPS改用HTTP。您可以使用以下示例策略作为指导：

```
{
  "Id": "ExamplePolicy",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublishThroughSSLOnly",
      "Action": "SNS:Publish",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:sns:us-east-1:1234567890:test-topic"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      },
      "Principal": "*"
    }
  ]
}
```

考虑使用VPC终端节点访问亚马逊 SNS

如果您有必须能够与之交互的主题，但这些主题绝对不能暴露在互联网上，请使用VPC端点将主题访问权限限制为仅限特定主机中的主机VPC。您可以使用主题策略来控制从特定 Amazon VPC 终端节点或特定终端节点访问主题的权限VPCs。

Amazon SNS VPC 终端节点提供了两种控制消息访问权限的方法：

- 您可以控制允许通过特定VPC端点的请求、用户或群组。
- 您可以使用主题策略控制哪些VPCs或VPC终端节点可以访问您的主题。

有关更多信息，请参阅创建终端节点 和为亚马逊创建亚马逊VPC终端节点策略 SNS。

确保订阅未配置为提供原始 http 终端节点

避免将订阅配置为传输到原始 http 终端节点。始终将订阅传输到终端节点域名。例如，配置为向终端节点 `http://1.2.3.4/my-path` 传输的订阅应该更改为 `http://my.domain.name/my-path`。

Amazon 疑难解答SNS主题

本节提供有关对 Amazon SNS 主题进行故障排除的信息。

使用 Amazon SNS 主题疑难解答 AWS X-Ray

AWS X-Ray 收集有关您的应用程序所处理的请求的数据，并允许您查看和筛选数据，以确定潜在的问题和优化机会。对于对您的应用程序的任何跟踪请求，您可以查看有关请求、响应以及应用程序对下游 AWS 资源、微服务、数据库和 HTTP Web APIs 的调用的详细信息。

您可以使用 X-Ray wit SNS h Amazon 来跟踪和分析通过您的应用程序传输的消息。您可以使用 AWS Management Console 来查看 Amazon SNS 与您的应用程序使用的其他服务之间的连接图。您还可以使用控制台查看指标，例如平均延迟和故障率。有关更多信息，请参阅 Amazon SNS 和 AWS X-Ray《AWS X-Ray 开发者指南》。

在 Amazon 上进行主动追踪 SNS

当用户通过您的亚马逊SNS主题传输 AWS X-Ray 到您的亚马逊数据 Firehos e、Amazon 和 HTTP/S 终端节点订阅时，AWS Lambda您可以使用来跟踪和分析他们的请求。SQS由 end-to-end 于 X-Ray 为您提供整个请求的视图，因此您可以查看您的 Amazon SNS 主题的名称以及主题订阅的下游内容。您可以分析消息及其后端服务的延迟（例如，请求在某个主题上花费了多长时间，以及将消息传送到该主题的每个订阅花费了多长时间）。

> **Important**
>
> 订阅量较多的 Amazon SNS 主题可能会达到大小限制且无法完全追踪。有关跟踪文档大小限制的信息，请参阅《 AWS 一般参考》中的 X 射线服务配额。

如果您通过已被追踪的服务致电亚马逊 SNSAPI，即使未在上启用 X-Ray 追踪，Amazon 也会SNS传递追踪信息API。

Amazon SNS 支持标准版和FIFO主题版的 X-Ray 跟踪。您可以使用亚马逊SNS控制台、亚马逊、亚马逊简单通知服务CLI参考或，为亚马逊SNSSNSSetTopicAttributesAPI主题启用 X-Ray AWS CloudFormation。

要详细了解如何将亚马逊SNS与 X-Ray 配合使用，请参阅亚马逊SNS和 AWS X-Ray《 AWS X-Ray 开发者指南》。

主题

主动跟踪权限

使用亚马逊SNS控制台时，亚马逊SNS会尝试为亚马逊SNS主题创建调用 X-Ray 所需的权限。如果您没有足够的权限使用Amazon SNS 控制台，则尝试可能会被拒绝。有关更多信息，请参阅Amazon 中的身份和访问管理 SNS 和Amazon SNS 访问控制的示例案例。

使用时CLI，必须手动配置权限。这些权限是使用资源策略配置的。有关在 X-Ray 中使用所需权限的更多信息，请参阅 Amazon SNS 和 AWS X-Ray。

使用控制台对 Amazon SNS 主题启用主动跟踪

在 Amazon SNS 主题上启用主动跟踪后，它会读取跟踪 ID，根据跟踪 ID 将数据发送给客户，并将跟踪 ID 传播到下游服务。

1. 登录 Amazon SNS 控制台。
2. 选择一个主题或创建一个新主题。有关创建主题的更多详情，请参阅创建 Amazon SNS 主题。
3. 在创建主题页面的详细信息部分，选择主题类型：FIFO或标准。
   a. 输入主题的名称。
   b. （可选）输入主题的显示名称。
4. 展开 Active tracing（主动跟踪），然后选择 Use active tracing（使用主动跟踪）。

为亚马逊SNS主题启用 X-Ray 后，您可以使用 X-Ray 服务地图查看该主题的 end-to-end 跟踪和服务地图。

使用 Amazon SNS 主题启用主动跟踪 AWS SDK

以下代码示例演示如何使用 AWS SDK适用于 Java 的，对亚马逊SNS主题启用主动跟踪。

```
public static void enableActiveTracing(SnsClient snsClient, String topicArn) {

        try {

            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                .attributeName("TracingConfig")
                .attributeValue("Active")
                .topicArn(topicArn)
                .build();

            SetTopicAttributesResponse result = snsClient.setTopicAttributes(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn()
                + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
```

使用 Amazon SNS 主题启用主动跟踪 AWS CLI

以下代码示例说明如何使用在 Amazon SNS 主题上启用主动跟踪 AWS CLI。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

使用在 Amazon SNS 主题上启用主动跟踪 AWS CloudFormation

以下 AWS CloudFormation 堆栈显示了如何在 Amazon SNS 主题上启用主动跟踪。

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
```

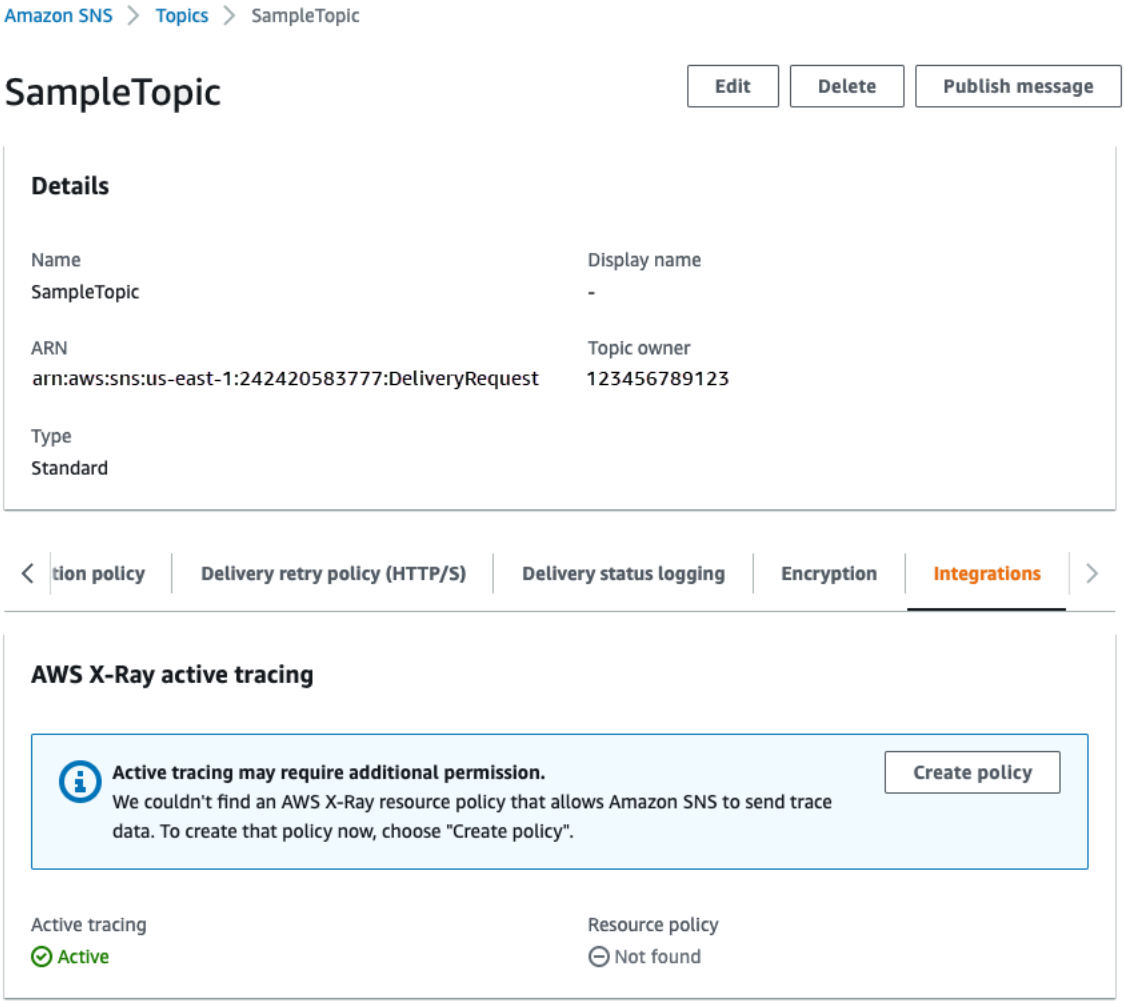
```
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

验证为您的主题启用了主动跟踪

您可以使用 Amazon SNS 控制台来验证您的主题是否启用了主动跟踪，或者资源策略添加失败。

1. 登录 Amazon SNS 控制台。
2. 在左侧导航窗格中，选择主题。
3. 在 Topics（主题）页上，选择一个主题。
4. 请选择 Integrations（集成）选项卡。

   启用主动跟踪后，会显示绿色的 Active（活动）图标。
5. 如果您启用了主动跟踪但没有看到资源策略已添加，请选择 Create policy（创建策略）以添加所需的额外权限。



测试主动跟踪

1. 登录 Amazon SNS 控制台。
2. 创建 Amazon SNS 主题。有关如何执行该操作的详细信息，请参阅 要使用创建主题 AWS Management Console。
3. 展开 Active tracing（主动跟踪），然后选择 Use active tracing（使用主动跟踪）。
4. 向 Amazon SNS 主题发布消息。有关如何执行该操作的详细信息，请参阅 要使用 Amazon SNS 主题发布消息 AWS Management Console。
5. 使用 X-Ray 服务地图查看该主题的 end-to-end 跟踪和服务地图。


51ms
0.2 t/min
https://sqs.us-east-1...
AWS::SQS::Queue
77ms
0.2 t/min
xray-lambda
AWS::Lambda
Client
11ms
0.2 t/min
xray-topic
AWS::SNS::Topic
387ms
0.2 t/min
remote
13ms
0.2 t/min
KinesisFirehose
AWS::KinesisFirehose

亚马逊SNS文件历史记录

下表介绍了对 Amazon Simple Notification Service 开发人员指南的最近更改。

服务功能有时会逐步推广到提供服务的 AWS 区域。我们仅在第一次发布时更新了此文档。我们不提供有关区域可用性的信息，也不会宣布后续区域支持情况。有关服务功能的区域可用性以及订阅更新通知的信息，请参阅新增内容 AWS？ 。

变更	说明	日期
加拿大西部（卡尔加里）对FIFO主题的支持	Amazon SNS 支持加拿大西部（卡尔加里）FIFO的主题。	2024 年 3 月 28 日
亚马逊在五个新区域SNSSMS提供支持	Amazon SNS 增加了对以下地区的SMS支持：亚太地区（海得拉巴）、亚太地区（墨尔本）、中东（UAE）、欧洲（苏黎世）和欧洲（西班牙）。	2024 年 2 月 8 日
Firebase 云消息传递 (FCM) HTTP v1 支持	亚马逊SNS支持 FCM v1 证书。	2024 年 1 月 18 日
亚太地区（雅加达）SNSSMS支持亚马逊	Amazon SNS 支持亚太地区（雅加达）的SMS消息传送。	2023 年 12 月 14 日
AWS CloudFormation 支持为 Amazon SNS 主题DeliveryStatusLogging 进行配置	AWS CloudFormation 支持在创建或更新 Amazon SNS 主题DeliveryStatusLogging 时进行配置。	2023 年 12 月 7 日
增加了新的消息筛选运算符	现在，在筛选 Amazon 消息时，您可以使用后缀匹配、等于忽略大小写和 OR 运算符。SNS	2023 年 11 月 16 日
增加了对消息归档与重播功能的支持	主题所有者可以将主题的消息归档最多 365 天。主题订阅用	2023 年 10 月 26 日

	户可以将归档的消息重播回订阅的端点，以恢复由于下游应用程序出现故障而受影响的消息，或者复制现有应用程序的状态。	
为标准队列订阅主题添加了 Support FIFO	您可以为亚马逊SQSFIFO队列或标准队列订阅亚马逊SNSFIFO主题。只有 Amazon SQS FIFO 队列才能保证消息按顺序接收，并且没有重复内容。	2023 年 9 月 14 日
SMS增加了对以色列（特拉维夫）的支持	以色列（特拉维夫）地区现在支持亚马逊SNSSMS。	2023 年 8 月 28 日
FIFO主题中增加了对 X-Ray 主动追踪的支持	以前仅支持亚马逊SNS标准主题， AWS X-Ray 现在可以跟踪和分析用户通过您的FIFO主题传送到您的 Amazon Data Firehose、Amaz AWS Lambda on 和 HTTP /S 终端节点SQS订阅的请求。	2023 年 5 月 31 日
增强的 Content-Type 标头支持	您可以在请求策略中设置 Content-Type 标头，以指定通知的媒体类型。	2023 年 3 月 23 日
添加了主动跟踪支持	AWS X-Ray 跟踪和分析用户通过您的亚马逊SNS标准主题传输到您的亚马逊数据 Firehose、 AWS Lambda Amazon 和 HTTP /S 终端节点SQS订阅的请求。	2023 年 2 月 8 日
新加坡发件人 ID 注册	添加了IDs在新加坡注册发件人的说明。	2023 年 1 月 10 日

基于有效负载的消息筛选	您可以利用基于有效负载的筛选来根据消息有效负载筛选消息，避免处理不需要的数据而产生相关成本。	2022 年 11 月 22 日
SHA256为 Amazon SNS 消息签名添加了哈希算法	使用 Amazon SNS 消息签名时增加了对SHA256哈希算法的支持。	2022 年 9 月 15 日
SMS消息传递中添加了其他区域	Amazon SNS 支持以下地区的SMS消息传送：非洲（开普敦）、亚太地区（大阪）、欧洲（米兰）和 AWS GovCloud（美国东部）。	2022 年 9 月 9 日
增加了消息数据保护支持	消息数据保护使用数据保护策略来审计和屏蔽在应用程序或 AWS 服务之间移动的敏感信息，从而保护发布到您的 Amazon SNS 主题的数据。	2022 年 9 月 8 日
免费电话号码的新注册流程	支持使用免费电话号码 (TFN) 向美国收件人发送 Amazon SNS 消息。	2022 年 8 月 1 日
支持基于属性的访问控制 () ABAC	为包括和在内的APIPublish操作添加了对基于属性的访问控制 (ABAC) 的支持。PublishBatch ABAC是一种授权策略，它根据标签定义访问权限，这些标签可以附加到IAM资源（例如IAM用户和角色）以及 AWS 资源（如 Amazon SNS 主题），以简化权限管理。	2022 年 1 月 10 日

支持推送通知的基于 Apple 令牌的身份验证	您可以授权亚马逊SNS向您的 iOS 或 macOS 应用程序发送推送通知，方法是提供可识别您是应用程序开发者的信息。	2021 年 10 月 28 日
新的SMS邮件发件人被放置在沙箱中 SMS	SMS沙盒的存在是为了帮助防止欺诈和滥用行为，并帮助保护您作为发件人的声誉。当您的 AWS 账户处于SMS沙箱中时，您只能向经过验证的目标电话号码发送SMS消息。	2021 年 6 月 1 日
新的SMS邮件发件人被放置在沙箱中 SMS	SMS沙盒的存在是为了帮助防止欺诈和滥用行为，并帮助保护您作为发件人的声誉。当您的 AWS 账户处于SMS沙箱中时，您只能向经过验证的目标电话号码发送SMS消息。	2021 年 6 月 1 日
向印度收件人发送SMS消息的新属性	现在需要两个新属性，即实体 ID 和模板 ID，才能向印度的收件人发送SMS消息。	2021 年 4 月 22 日
消息筛选运算符的更新	新的运算符 `cidr` 可用于匹配消息源 IP 地址和子网。您现在还可以检查是否存在属性键，并使用前缀与 `anything-but` 运算符进行属性字符串匹配。	2021 年 4 月 7 日

终止对美国目的地的 P2P 长代码的支持	自 2021 年 6 月 1 日起，美国电信提供商不再支持使用 person-to-person (P2P) 长码与美国目的地通信 application-to-person (A2P)。取而代之的是，您可以使用短码、免费电话号码或名为 10 的新型发起号码。DLC	2021 年 2 月 16 日
支持 1 分钟 Amazon 指标 CloudWatch	Amazon SNS 的 1 分钟 CloudWatch 指标现已在所有 AWS 地区推出。	2021 年 1 月 28 日
支持 Amazon Data Firehose 终端节点	您可以为 SNS Firehose 直播订阅主题。这允许您向存档和分析终端节点发送通知，例如亚马逊简单存储服务 (Amazon S3) 存储桶、Amazon Redshift 表、 OpenSearch 亚马逊服务 OpenSearch （ 服务 ）等。	2021 年 1 月 12 日
源号码可用	发送短信时可以使用发件人号码 (SMS)。	2020 年 10 月 23 日
Support for Amazon SNS FIFO 主题	要集成需要近乎实时的数据一致性的分布式应用程序，您可以将 Amazon 先入SNS先出 (FIFO) 主题与 Amazon 队列一起使用。SQS FIFO	2020 年 10 月 22 日
适用于 Java 的 Amazon SNS 扩展客户端库现已推出	您可以使用此库发布大型 Amazon SNS 消息。	2020 年 8 月 25 日
SSE已在中国地区上市	Amazon SNS 的服务器端加密 (SSE) 在中国地区可用。	2020 年 1 月 20 日

Support 支持DLQs使用捕获无法投递的邮件	要捕获无法送达的消息，您可以将亚马逊SQS死信队列 () DLQ 与亚马逊订阅一起使用。SNS	2019 年 11 月 14 日
Support 支持指定自定义APNs标题值	您可以指定自定义标APNs题值。	2019 年 10 月 18 日
Support 支持 apns-push-type " 标题字段 APNs	您可以将标apns-push-type题字段用于通过发送的移动通知APNs。	2019 年 9 月 10 日
Support 支持使用主题疑难解答 AWS X-Ray	您可以使用 X-Ray 对通过SNS主题传递的消息进行故障排除。	2019 年 7 月 24 日
支持使用“exists”运算符进行属性键匹配	要检查传入消息中是否存在其键与筛选策略中列出的键匹配的属性，您可以使用 exists 运算符。	2019 年 7 月 5 日
支持多个数值的 anything-but 匹配	除了多个字符串外，Amazon还SNS允许对多个数值进行除匹配之外的任何操作。	2019 年 7 月 5 日
Amazon SNS 发行说明以RSS提要形式提供	在此页面的标题（文档历史记录）之后，选择RSS。	2019 年 6 月 22 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。