aws

開発者ガイド

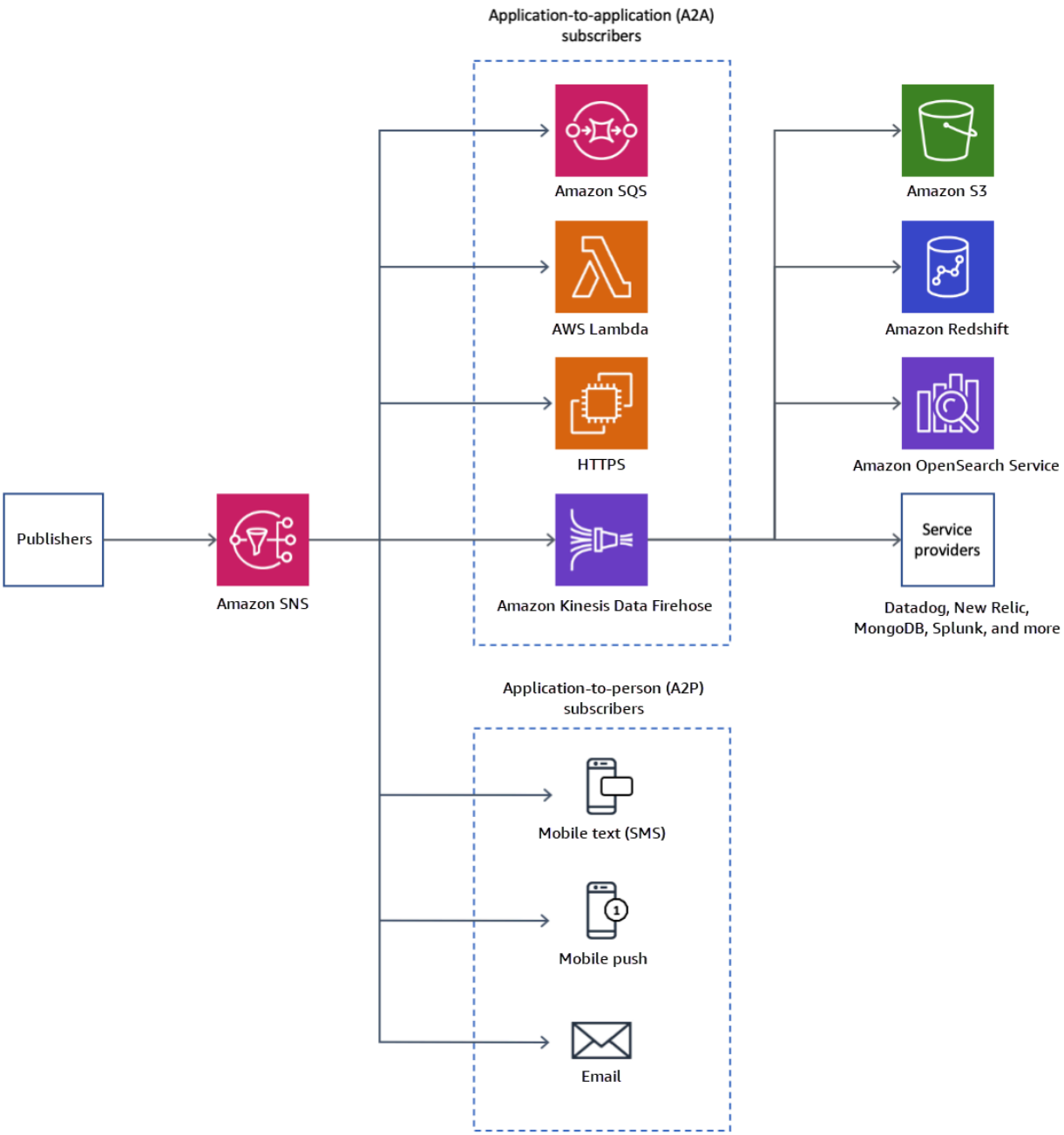
Amazon Simple Notification Service

Amazon Simple Notification Service: 開発者ガイド

Table of Contents

Amazon SNS とは

Amazon Simple Notification Service (Amazon SNS) は、配信者から受信者 (または生産者から消費者) へのメッセージ配信を提供するマネージドサービスです。発行者は、論理アクセスポイントおよび通信チャネルであるトピックにメッセージを送信することで、受信者と非同期的に通信します。クライアントは SNS トピックをサブスクライブし、Amazon Data Firehose、Amazon SQS、HTTP、E メール AWS Lambda、モバイルプッシュ通知、モバイルテキストメッセージ (SMS) など、サポートされているエンドポイントタイプを使用して公開されたメッセージを受信できます。



トピック

- 特徴と機能
- 関連サービス
- Amazon SNS へアクセスする

- Amazon SNS の料金
- 一般的な Amazon SNS シナリオ
- AWS SDK での Amazon SNS の使用

特徴と機能

Amazon SNS には、以下の特徴と機能があります。

- pplication-to-application メッセージング

  pplication-to-application メッセージングは、Amazon Data Firehose 配信ストリーム、Lambda 関数、Amazon SQS キュー、HTTP/S エンドポイント、 AWS Event Fork Pipelines などのサブスクライバーをサポートします。詳細については、「アプリケーショントゥーアプリケーション (A2A) メッセージング」を参照してください。

- pplication-to-person 通知

  pplication-to-person 通知は、モバイルアプリケーション、携帯電話番号、E メールアドレスなどのサブスクライバーにユーザー通知を提供します。詳細については、「Application-to-Person (A2P) メッセージング」を参照してください。

- 標準および FIFO トピック

  FIFO トピックを使用して、メッセージの順序を厳格に保ち、メッセージグループを定義し、メッセージの重複を防止します。FIFO トピックにサブスクライブするには、FIFO キューと標準キューの両方を使用できます。詳細については、「メッセージの順序付けと重複除外 (FIFO トピック)」を参照してください。

  メッセージの配信順序やメッセージの重複が重要でない場合は、標準トピックを使用します。サポートされているすべての配信プロトコルは、スタンダードトピックにサブスクライブできます。

- メッセージの耐久性

  Amazon SNS では、メッセージの耐久性を実現するために連携するいくつかの戦略を使用しています。

  - 公開されたメッセージは、地理的に分離された複数のサーバーおよびデータセンターの間で保存されます。
  - サブスクライブされたエンドポイントが利用できない場合、Amazon SNS は配信の再試行ポリシーを実行します。

  - 配信再試行ポリシーが終了する前に配信されなかったメッセージを保持するには、デッドレターキューを作成できます。
- メッセージのアーカイブ、リプレイ、および分析

  Amazon SNS でメッセージをアーカイブするには、Firehose 配信ストリームを SNS トピックにサブスクライブします。これにより、Amazon Simple Storage Service (Amazon S3) バケット、Amazon Redshift テーブルなどの分析エンドポイントに通知を送信できます。また、Amazon SNS FIFO トピックは、トピック所有者がトピック内でメッセージを保存 (またはアーカイブ) できる、ノーコードのインプレースメッセージアーカイブとしてメッセージのアーカイブとリプレイをサポートしています。トピックサブスクライバーは、アーカイブされたメッセージをサブスクライブされたエンドポイントに対してリトリーブ (またはリプレイ) できます。詳細については、「FIFO トピックのメッセージのアーカイブとリプレイ」を参照してください。
- メッセージ属性

  メッセージ属性により、メッセージに関する構造化メタデータを提供することができます。the section called “メッセージ属性”。
- メッセージのフィルター処理

  デフォルトでは、 各受信者は、トピックに対して発行されたすべてのメッセージを受信します。メッセージのサブセットを受信する場合、受信者はトピックのサブスクリプションにフィルターポリシーを割り当てる必要があります。サブスクライバーはフィルターポリシーの範囲を定義して、ペイロードベースまたは属性ベースのフィルタリングを有効にすることもできます。フィルターポリシー範囲のデフォルト値は `MessageAttributes` です。受信メッセージ属性がフィルターポリシー属性と一致すると、メッセージはサブスクライブされたエンドポイントに配信されます。それ以外の場合、メッセージは除外されます。フィルターポリシーの範囲が `MessageBody` の場合、フィルターポリシー属性がペイロードと照合されます。詳細については、「メッセージのフィルター処理」を参照してください。
- メッセージセキュリティ

  サーバー側の暗号化は、 が提供する暗号化キーを使用して、Amazon SNS トピックに保存されているメッセージの内容を保護します AWS KMS。詳細については、「the section called “保管中の暗号化”」を参照してください。

  Amazon SNS と Virtual Private Cloud (VPC) の間にプライベート接続を確立することもできます。詳細については、「the section called “インターネットトラフィックのプライバシー”」を参照してください。

関連サービス

Amazon SNS では、以下のサービスを使用できます。

- Amazon SQS は、分散されたソフトウェアシステムとコンポーネントを統合および疎結合化できる、安全性、耐久性、可用性に優れたホストされたキューを提供します。Amazon SQS は、以下で Amazon SNS に関連しています。
  - Amazon SNS では、Amazon SQS による、配信不能なメッセージのためのデッドレターキューを提供します。
  - Amazon SQS キューを Amazon SNS トピックにサブスクライブできます。
  - Amazon SQS FIFO キューまたは標準キューを Amazon SNS FIFO トピックにサブスクライブできます。Amazon SQS FIFO キューに限り、メッセージが順番に受信され、重複がないことが保証されます。
- AWS Lambda により、新規情報に迅速に対応するアプリケーションを構築できます。可用性に優れたコンピューティングインフラストラクチャで Lambda 関数のアプリケーションコードを実行します。詳細については、AWS Lambda デベロッパーガイドを参照してください。Lambda 関数を SNS トピックにサブスクライブできます。
- AWS Identity and Access Management (IAM) は、ユーザーの AWS リソースへのアクセスを安全に制御するのに役立ちます。IAM により、どのユーザーがお客様の Amazon SNSトピックを使用できるか (認証)、それらのユーザーがどのリソースをどのような方法で使用できるか (承認) をコントロールできます。詳細については、「Amazon SNS でのアイデンティティベースのポリシーを使用する」を参照してください。
- AWS CloudFormation では、 AWS リソースをモデル化してセットアップできます。Amazon SNS トピックやサブスクリプションなど、必要な AWS リソースを記述するテンプレートを作成します。 AWS CloudFormation は、これらのリソースのプロビジョニングと設定を処理します。詳細については、『AWS CloudFormation ユーザーガイド』を参照してください。

Amazon SNS へアクセスする

Amazon SNS コンソール、コマンドラインツール、または AWS SDKs Amazon SNS トピックとサブスクリプションを設定および管理できます。

- Amazon SNS コンソールは、トピックとサブスクリプションの作成、メッセージの送信と受信、イベントとログのモニタリングを行うための便利なユーザーインターフェイスを提供します。

- AWS Command Line Interface （ AWS CLI) を使用すると、高度な設定と自動化のユースケースのために Amazon SNS API に直接アクセスできます。詳細については、「AWS CLIでの Amazon SNS の使用」を参照してください。
- AWS は、さまざまな言語で SDKsを提供します。詳細については、 「SDK とツールキット」を参照してください。

Amazon SNS の料金

Amazon SNS には前払いのコストはかかりません。発行するメッセージの数、配信する通知の数、およびトピックとサブスクリプションを管理するための追加の API コールに基づいてお支払いいただきます。配信料金は、エンドポイントのタイプによって異なります。Amazon SNS 無料利用枠を使用すれば、無料で始められます。

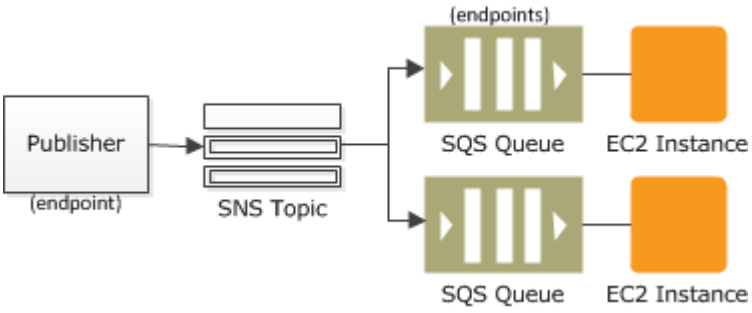
詳細については、「Amazon SNS の料金」を参照してください。

一般的な Amazon SNS シナリオ

アプリケーション統合

ファンアウトシナリオは、SNS トピックに発行されたメッセージがレプリケートされ、Firehose 配信ストリーム、Amazon SQS キュー、HTTP(S) エンドポイント、Lambda 関数などの複数のエンドポイントにプッシュされるときです。これで並列非同期処理が可能になります。

例えば、ある製品が注文されるたびにトピックに SNS メッセージを送信するアプリケーションを開発できます。そうすると、その SNS トピックにサブスクライブされている SQS キューは、新しい注文に関する同一の通知を受け取ります。SQS キューの 1 つに添付された Amazon Elastic Compute Cloud (Amazon EC2) サーバーインスタンスは、注文の処理や実行を処理できます。また、受信したすべての注文を分析するために、別の Amazon EC2 サーバーインスタンスをデータウェアハウスにアタッチすることもできます。



ファンアウトは、本番稼働用環境に送信されるデータを、テスト環境にレプリケートすることにも使用できます。前の例から発展させて、新しい受注用に同じ SNS トピックに別の SQS キューをサブスクライブすることもできます。次に、この新しい SQS キューをテスト環境に添付することで、本番稼働用環境から受け取ったデータを使用してアプリケーションの改善とテストを継続することができます。

> **Important**
>
> 本番稼働用データをテスト環境に送信する前に、データのプライバシーとセキュリティを考慮してください。

詳細については、以下のリソースを参照してください。

- Firehose 配信ストリームへのファンアウト
- Lambda 関数へのファンアウト
- Amazon SQS キューへのファンアウト
- HTTP(S) エンドポイントへのファンアウト
- Amazon SNS と AWS コンピューティング、ストレージ、データベース、ネットワークサービスによるイベント駆動型コンピューティング

アプリケーションアラート

アプリケーションおよびシステムアラートは、事前定義されたしきい値によってトリガーされた送信される通知です。Amazon SNS は、SMS および E メールを使用して、指定されたユーザーにこれらの通知を送信できます。例えば、Amazon EC2 Auto Scaling グループへの特定の変更、Amazon S3 バケットにアップロードされた新しいファイル、Amazon でメトリクスのしきい値を超えたなど、イベントが発生したときにすぐに通知を受け取ることができます CloudWatch。詳細については、Amazon SNS 通知の設定 CloudWatch 」を参照してください。

ユーザー通知

Amazon SNS では、プッシュ E メールメッセージとテキストメッセージ (SMS メッセージ) を個人またはグループに送信できます。例えば、e コマースの注文確認をユーザー通知として送信できます。Amazon SNS を使用した SMS メッセージの送信の詳細については、「モバイルテキストメッセージング (SMS)」を参照してください。

モバイルプッシュ通知

モバイルプッシュ通知を使用すると、モバイルアプリケーションに直接メッセージを送信できます。例えば、Amazon SNS を使用して、アプリケーションに更新通知を送信できます。通知メッセージには、更新をダウンロードおよびインストールするためのリンクを含めることができます。Amazon SNS を使用したプッシュ通知メッセージの送信の詳細については、「モバイルプッシュ通知」を参照してください。

AWS SDK での Amazon SNS の使用

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	PowerShell コード例のツール
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例

SDK ドキュメント	コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

Amazon SNS 固有の例については、「SDK を使用した Amazon SNS のコード例 AWS SDKs」を参照してください。

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

Amazon SNS イベントのソースと送信先

Amazon SNS は、多くの AWS ソースからのイベント駆動型通知と、Application-to-Application (A2A) と Application-to-Person (A2P) 送信先へのファンアウト通知を受信できます。このセクションでは、サポートされているイベントの送信元と送信先の一覧と、詳細情報へのリンクを表示しています。

トピック

Amazon SNS イベントソース

このページでは、Amazon SNS トピックにイベントを発行できる AWS のサービスをAWS 製品カテゴリごとにグループ分けして一覧で表示しています。

> **Note**
>
> 2020 年 10 月に Amazon SNS は FIFO トピックを導入しました。現在、ほとんどの AWS サービスは、標準トピックへのイベントの送信のみをサポートします。

分析サービス

AWS のサービス	Amazon SNS を使用する利点
Amazon Athena - 標準 SQL を使用した Amazon S3 内のデータの分析ができます。	制御制限を超えたときに通知を受信します。詳細については、『Amazon Athena ユーザーガイド』の「データ使用量管理制限の設定」を参照してください。
AWS Data Pipeline - データの移動と変換を自動化するのに役立ちます。	パイプラインコンポーネントのステータスに関する通知を受信します。詳細については、「AWS Data Pipeline デベロッパーガイド」の「SnsAlarm」を参照してください。

AWS のサービス	Amazon SNS を使用する利点
Amazon Redshift - データウェアハウスを設定、操作、および拡張するためのすべての作業を管理します。	Amazon Redshift イベントの通知を受信します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift イベント通知」を参照してください。

アプリケーション統合サービス

AWS のサービス	Amazon SNS を使用する利点
Amazon EventBridge – 独自のアプリケーション、 software-as-a-service (SaaS) アプリケーション、および AWSのサービスからリアルタイムデータのストリームを配信し、そのデータを Amazon SNS . EventBridge was の以前は CloudWatch Events と呼ばれていたターゲットにルーティングします。	EventBridge イベントの通知を受信します。詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon ターゲット EventBridge 」を参照してください。
AWS Step Functions - AWS Lambda 機能と他の AWS サービスを組み合わせて、ビジネスクリティカルなアプリケーションを構築できます。	Step Functions イベントの通知を受信します。詳細については、『AWS Step Functions デベロッパーガイド』の「Step Functions で Amazon SNS を呼び出す」を参照してください。

請求情報とコスト管理サービス

AWS のサービス	Amazon SNS を使用する利点
AWS Billing and Cost Management - コストと請求書の支払いのモニタリングを支援する機能を提供します。	予算通知、料金変更通知、異常アラートを受信します。詳細については、「AWS Billing ユーザーガイド」の次のページを参照してください。 •

AWS のサービス	Amazon SNS を使用する利点
	予算の通知に関する Amazon SNS トピックを作成する • 通知の設定 • AWS コスト異常検出で異常な支出を検出する

ビジネスアプリケーションサービス

AWS のサービス	Amazon SNS を使用する利点
Amazon Chime - 組織の内外を問わず、会い、チャットをし、仕事の電話をかけられます。	重要な会議イベント通知を受信します。詳細については、『Amazon Chime デベロッパーガイド』の「Amazon Chime SDK イベント」を参照してください。

コンピューティングサービス

AWS のサービス	Amazon SNS を使用する利点
Amazon EC2 Auto Scaling - アプリケーションの負荷を処理する正しい数の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを確保できます。	Auto Scaling が Auto Scaling グループの Amazon EC2 インスタンスを起動または終了するときに、通知を受信します。詳細については、『Amazon EC2 Auto Scaling ユーザーガイド』の「Auto Scaling グループのスケーリング時の Amazon SNS 通知の取得」を参照してください。
EC2 Image Builder – 特定の IT 標準を満たすためにソフトウェアと設定が事前にインストールおよび設定された、カスタマイズされた安全な	構築が完了すると通知を受信します。詳細については、『AWS コンピューティングのブログ』の「EC2 Image Builder パイプラインでの

AWS のサービス	Amazon SNS を使用する利点
サーバーイメージの作成、 up-to-date管理、デプロイを自動化するのに役立ちます。	最新のサーバーイメージの追跡」を参照してください。
AWS Elastic Beanstalk - アプリケーションのキャパシティープロビジョニング、負荷分散、スケーリング、およびアプリケーションの状態のモニタリングといった詳細を処理します。	アプリケーションに影響を与える重要なイベントの通知を受信します。詳細については、『AWS Elastic Beanstalk デベロッパーガイド』の「Amazon SNS を使用した Elastic Beanstalk 環境通知」を参照してください。
AWS Lambda - サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。	SNS トピックを Lambda デッドレターキューまたは Lambda の送信先として設定することで、関数出力データを受信します。詳細については、『AWS Lambda デベロッパーガイド』の「非同期の呼び出し」を参照してください。
Amazon Lightsail - デベロッパーが AWS を使用して ウェブサイトや ウェブアプリケーション構築に着手する手助けをします。	インスタンス、データベース、ロードバランサーの 1 つのメトリクスが指定したしきい値を超えた場合に通知を受信します。詳細については、『Amazon Lightsail デベロッパーガイド』の「Amazon Lightsail での通知連絡先の追加」を参照してください。

コンテナサービス

AWS のサービス	Amazon SNS を使用する利点
Amazon EKS Distro - アプリケーションがデプロイされている場所に関係なく、信頼性の高い安全なクラスターを作成できます。	Amazon EKS Distro で作成されたクラスターの更新とセキュリティパッチを追跡します。詳細については、「Amazon EKS Distro - Amazon EKS で使用されるオープンソースの Kubernetes ディストリビューション」を参照してください。

AWS のサービス	Amazon SNS を使用する利点
Amazon Elastic Container Service (Amazon ECS) - クラスターでコンテナを実行、停止、管理できます。	新しい Amazon ECS 最適化 AMI が利用可能になったときに通知を受信します。詳細については、『Amazon Elastic Container Service デベロッパーガイド』の「Amazon ECS に最適化された AMI 更新通知にサブスクライブする」を参照してください。

カスタマーエンゲージメントサービス

AWS のサービス	Amazon SNS を使用する利点
Amazon Connect - オムニチャネルクラウドコンタクトセンターをセットアップして、顧客とやり取りすることができます。	アラートと検証を受信します。詳細については、『Amazon Connect 管理者ガイド』の「Amazon Connect を使用した AWS の性能」を参照してください。
Amazon Pinpoint- E メール、SMS、音声メッセージ、プッシュ通知を送信することで、顧客のエンゲージメントを支援します。	双方向 SMS を構成しています。これにより、顧客からメッセージを受信することができます。詳細については、『Amazon Pinpoint ユーザーガイド』の「Amazon Pinpoint での双方向 SMS メッセージングを使用する」を参照してください。
Amazon Simple Email Service (Amazon SES) - ユーザー自身の E メールアドレスとドメインを使用して E メールを送受信するための、費用対効果の高い方法を提供します。	バウンス、苦情および配信に関する通知を受信します。詳細については、『Amazon Simple Email Service デベロッパーガイド』の、「Amazon SES のための Amazon SNS 通知を設定する」を参照してください。

データベースサービス

AWS のサービス	Amazon SNS を使用する利点
AWS Database Migration Service - オンプレミスのデータベースから AWS クラウドへデータを移行します。	例えば、レプリケーションインスタンスが作成または削除された場合などの AWS DMS イベントが発生すると通知を受信します。詳細については、『AWS Database Migration Service ユーザーガイド』の「AWS Database Migration Service のイベントと通知を使用する」を参照してください。
Amazon DynamoDB - シームレスな拡張性により、フルマネージドの NoSQL データベースサービスの高速で予測可能なパフォーマンスを提供します。	メンテナンスイベントが発生したときに通知を受信します。詳細については、『Amazon DynamoDB デベロッパーガイド』の「DAX クラスターの設定をカスタマイズする」を参照してください。
Amazon ElastiCache — 高パフォーマンスでサイズ変更可能でコスト効率の高いインメモリキャッシュを提供し、分散キャッシュ環境のデプロイと管理に伴う複雑さを排除します。	重要なイベントが発生したときに通知を受信します。詳細については、「Amazon for Memcached ユーザーガイド」の「イベント通知と Amazon SNS」を参照してください。ElastiCache
Amazon Neptune - 高度に接続されたデータセットを扱うアプリケーションを構築し実行できます。	Neptune イベントが発生したときに通知を受信します。詳細については、『Neptune ユーザーガイド』の「Neptune イベント通知を使用する」を参照してください。
Amazon Redshift - データウェアハウスを設定、操作、およびスケーリングするためのすべての作業を管理します。	Amazon Redshift イベントの通知を受信します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift イベント通知」を参照してください。
Amazon Relational Database Service - AWS クラウドでリレーショナルデータベースを簡単に	Amazon RDS イベントの通知を受信します。詳細については、『Amazon RDS ユーザーガ

AWS のサービス	Amazon SNS を使用する利点
セットアップし、操作し、スケーリングすることができます。	イド』の「Amazon RDS イベント通知を使用する」を参照してください。

デベロッパー用ツールサービス

AWS のサービス	Amazon SNS を使用する利点
AWS CodeBuild - ソースコードをコンパイルし、単体テストを実行して、すぐにデプロイできるアーティファクトを生成します。	構築が成功したとき、失敗したとき、または構築フェーズから別の構築フェーズに移動したときに通知を受信します。詳細については、「 ユーザーガイド」の「 のビルド通知サンプル CodeBuildAWS CodeBuild」を参照してください。
AWS CodeCommit - クラウドでアセットを個人的に保存および管理するためのバージョン管理を提供します。	CodeCommit リポジトリイベントに関する通知を受信します。詳細については、『AWS CodeCommit ユーザーガイド』の「例: Amazon SNS トピック用 AWS CodeCommit トリガーを作成する」を参照してください。
AWS CodeDeploy - Amazon EC2 インスタンス、オンプレミスインスタンス、サーバーレス Lambda 関数、または Amazon ECS サービスに対するアプリケーションのデプロイを自動化します。	CodeDeploy デプロイまたはインスタンスイベントの通知を受信します。詳細については、「 ユーザーガイド」の CodeDeploy 「イベントのトリガーを作成するAWS CodeDeploy」を参照してください。
Amazon CodeGuru – ライブアプリケーションからランタイムパフォーマンスデータを収集し、アプリケーションのパフォーマンスを微調整するのに役立つレコメンデーションを提供します。	異常が発生したときに通知を受信します。詳細については、「Amazon CodeGuru ユーザーガイド」の「異常レポートとレコメンデーションレポートの使用」を参照してください。
AWS CodePipeline - ソフトウェアの変更を継続的にリリースするために必要なステップを自動化します。	承認アクションに関する通知を受信します。詳細については、「 ユーザーガイド」の「 で

AWS のサービス	Amazon SNS を使用する利点
	の承認アクションの管理 CodePipelineAWS CodePipeline」を参照してください。
AWS CodeStar – AWS 上のソフトウェア開発プロジェクトを作成、管理、および使用する。	使用するリソースで発生するイベントに関する通知を受信します。詳細については、『デベロッパー用ツールコンソールユーザーガイド』の「通知用 Amazon SNS トピックの設定」を参照してください。

フロントエンドウェブおよびモバイルサービス

AWS のサービス	Amazon SNS を使用する利点
Amazon Pinpoint - E メール、SMS、音声メッセージ、プッシュ通知を送信することで、顧客のエンゲージメントを支援します。	双方向 SMS を構成しています。これにより、顧客からメッセージを受信することができます。詳細については、『Amazon Pinpoint ユーザーガイド』の「Amazon Pinpoint で双方向 SMS メッセージングを使用する」を参照してください。

ゲーム開発サービス

AWS のサービス	Amazon SNS を使用する利点
Amazon GameLift — ゲームサーバーのデプロイ、運用、スケーリングのためのフルマネージドサービスを含む、セッションベースのマルチプレイヤーゲームサーバーをクラウドでホストするためのソリューションを提供します。	マッチメイキングとキューイベント通知を受信します。詳細については、次のページを参照してください。 • マッチメーキング通知については、「Amazon GameLift FlexMatch デベロッパーガイド」の FlexMatch 「イベント通知の設定」を参照してください。 •

AWS のサービス	Amazon SNS を使用する利点
	キュー通知については、「Amazon GameLift デベロッパーガイド」の「ゲームセッション配置のイベント通知を設定する」を参照してください。

IoT サービス

AWS のサービス	Amazon SNS を使用する利点
AWS IoT Core - IoT デバイスを他のデバイスや AWS クラウドサービスに接続するクラウドサービスを提供します。	AWS IoT Core イベントの通知を受信します。詳細については、『AWS IoT Amazon SNS デベロッパーガイド』の「トピックの作成」を参照してください。
AWS IoT Device Defender - デバイスの設定の監査、異常動作の検出を目的としたコネクテッドデバイスのモニタリング、セキュリティリスクの軽減ができます。	デバイスが動作に違反したときにアラームを受信します。詳細については、『AWS IoT デベロッパーガイド』の「AWS IoT Device Defender 検出の使用方法」を参照してください。
AWS IoT Events - 機器やデバイスフリートの障害や動作の変化をモニタリングし、そのようなイベントが発生したときにアクションをトリガーできます。	AWS IoT Events イベントの通知を受信します。詳細については、『AWS IoT Events デベロッパーガイド』の「Amazon Simple Notification Service」を参照してください。
AWS IoT Greengrass - AWS を物理的デバイスに拡張することにより管理、分析、耐久性のあるストレージのためにクラウドを使用しながら、生成したデータに基づいてローカルで動作することが可能になります。	AWS IoT Greengrass イベントの通知を受信します。詳細については、『AWS IoT Greengrass Version 1 デベロッパーガイド』の「SNS コネクター」を参照してください。

機械学習サービス

AWS のサービス	Amazon SNS を使用する利点
Amazon CodeGuru – ライブアプリケーションからランタイムパフォーマンスデータを収集し、アプリケーションのパフォーマンスを微調整するのに役立つレコメンデーションを提供します。	異常が発生したときに通知を受信します。詳細については、「Amazon CodeGuru ユーザーガイド」の「異常レポートとレコメンデーションレポートの使用」を参照してください。
Amazon DevOps Guru – 機械学習を使用して運用上のインサイトを生成し、運用アプリケーションのパフォーマンスを向上させます。	インサイトと確認を転送します。詳細については、「 AWS管理とガバナンスブログ」の「Amazon DevOps Guru PagerDuty を使用して ML を活用した運用上のインサイトをオンコールチームに配信する」を参照してください。
Amazon Lookout for Metrics - データ内の異常を検出し、その根本原因を特定し、迅速な対応を可能にします。	異常の通知を受信します。詳細については、『Lookout for Metrics で Amazon SNS を使用する』の「Amazon Lookout for Metrics デベロッパーガイド」を参照してください。
Amazon Rekognition - イメージ分析とビデオ分析をアプリケーションに追加できます。	リクエスト結果の通知を受信します。詳細については、『Amazon Rekognition デベロッパーガイド』の「リファレンス: ビデオ分析結果の通知」を参照してください。
Amazon SageMaker — データサイエンティストとデベロッパーが機械学習モデルを構築してトレーニングし、それらを本番環境に対応したホスト環境に直接デプロイできるようにします。	データオブジェクトがラベル付けされると、通知を受信します。詳細については、「Amazon SageMaker デベロッパーガイド」の「ストリーミングラベル付けジョブの作成」を参照してください。

管理とガバナンスサービス

AWS のサービス	Amazon SNS を使用する利点
AWS Chatbot — DevOps とソフトウェア開発チームが Amazon Chime と Slack チャットルームを使用して、 AWS クラウド内の運用イベントをモニタリングし、対応できるようにします。	チャットルームに通知を配信します。詳細については、『AWS Chatbot 管理者ガイド』の「AWS Chatbot のセットアップ」を参照してください。
AWS CloudFormation - 計画に従い、繰り返し AWS インフラストラクチャデプロイを作成し、プロビジョニングします。	スタックが作成され、更新されたときに通知を受信します。詳細については、『AWS CloudFormation ユーザーガイド』の「AWS CloudFormation スタックオプションの設定をする」を参照してください。
AWS CloudTrail - お客様の AWS アカウント アクティビティイベント履歴を提供します。	が新しいログファイルを Amazon S3 バケットに CloudTrail 発行するときに通知を受信します。詳細については、「 AWS CloudTrailユーザーガイド」の「 の Amazon SNS 通知の設定 CloudTrail」を参照してください。
Amazon CloudWatch – AWSリソースと AWS で実行しているアプリケーションをリアルタイムでモニタリングします。	アラームの変化状態に応じて通知を受信します。詳細については、「Amazon ユーザーガイド」の「Amazon CloudWatch アラームの使用」を参照してください。 CloudWatch
AWS Config - AWS アカウント にある AWS リソースの設定詳細ビューを提供します。	リソースが更新されたとき、または AWS Config がリソースに対しカスタムルールまたはマネージドルールを評価したときに通知を受信します。詳細については、『AWS Config デベロッパーガイド』の「AWS Config がSNS トピックに送信する通知」および「構成項目変更通知の例」を参照してください。
AWS Control Tower - 安全かつ基準に準拠した複数アカウント AWS 環境をセットアップおよび管理することができます。	アラートを使用すると、ランディングゾーン内のドリフトを防止し、コンプライアンス通知を受信できます。詳細については、『AWS

AWS のサービス	Amazon SNS を使用する利点
	Control Tower ユーザーガイド』の「Amazon Simple Notification Service でアラートを追跡する」を参照してください。
AWS License Manager - ソフトウェアベンダーのソフトウェアライセンスを AWS およびオンプレミス環境をまたいで一元的に管理するのに役立ちます。	License Manager の通知とアラートを受信します。詳細については、「License Manager ユーザーガイド」の「License Manager の設定」およびAWS「管理とガバナンスブログ」のAWS License Manager「通知用の ServiceNow インシデントの作成」を参照してください。
AWS Service Catalog - IT 管理者が承認された製品のポートフォリオを作成および管理し、そのポートフォリオをエンドユーザーに配布できます。エンドユーザーは、パーソナライズされたポータルから必要な製品にアクセスできます。	スタックイベントに関する通知を受信します。詳細については、「Service Catalog 管理者ガイド」の「AWS Service Catalog 通知の制約」を参照してください。
AWS Systems Manager - AWS のインフラストラクチャの表示と制御ができます。	コマンドのステータスに関する通知を受け取ります。詳細については、『AWS Systems Manager ユーザーガイド』の「Amazon SNS 通知を使用した Systems Manager のステータス変更のモニタリング」を参照してください。

メディアサービス

AWS のサービス	Amazon SNS を使用する利点
Amazon Elastic Transcoder - Amazon S3 に保存したメディアファイルをユーザーの再生デバイスに対応した形式のメディアファイルに変換できます。	ジョブのステータスが変更されたときに通知を受信します。詳細については、『Amazon Elastic Transcoder デベロッパーガイド』の「ジョブステータスの通知」を参照してください。

移行と転送のサービス

AWS のサービス	Amazon SNS を使用する利点
AWS Application Discovery Service - オンプレミスサーバーに関する使用状況と設定のデータを収集することで、AWS クラウドへの移行を計画できます。	AWS CloudTrail を介してイベントの通知を受信します。詳細については、『Application Discovery Service ユーザーガイド』の「AWS CloudTrail を使用したApplication Discovery Service API コールのログ記録」を参照してください。
AWS Database Migration Service - オンプレミスデータベースから AWS クラウドからデータを移行します。	例えば、レプリケーションインスタンスが作成または削除された場合などの AWS DMS イベントが発生すると通知を受信します。詳細については、『AWS Database Migration Service ユーザーガイド』の「AWS Database Migration Service のイベントと通知を使用する」を参照してください。
AWS Snowball — 物理ストレージデバイスを使用して、Amazon S3 とオンサイトのデータストレージロケーション間で大量のデータを faster-than-internet 高速に転送します。	Snowball ジョブの通知を受信します。詳細については、次を参照してください。 • 『AWS Snowball ユーザーガイド』の「Snowball 通知」 • ステップ 5: 「AWS Snowball Edge デベロッパーガイド」の通知設定を選択します • ステップ 5: 「AWS Snowball ユーザーガイド」の通知設定を選択します

ネットワーキングとコンテンツ配信サービス

AWS のサービス	Amazon SNS を使用する利点
Amazon API Gateway – 独自の REST および WebSocket APIs をあらゆる規模で作成およびデプロイできます。	API Gateway エンドポイントに投稿されたメッセージを受信します。詳細については、『API Gateway デベロッパーガイド』の「AWS を使用したチュートリアル: API Gateway REST API の構築」を参照してください。
Amazon CloudFront – .html、.css、.php、イメージ、メディアファイルなど、静的および動的なウェブコンテンツの配信を高速化します。	指定された CloudFront メトリクスに基づいてアラームが発生したときに通知を受信します。詳細については、「Amazon CloudFront デベロッパーガイド」の「通知を受け取るようにアラームを設定する」を参照してください。
AWS Direct Connect - お客様の内部ネットワークを AWS Direct Connect ロケーションに、標準のイーサネット光ファイバケーブルを介して接続します。	AWS Direct Connect接続変更状態をモニタリングするアラームが発生すると通知を受信します。詳細については、「 ユーザーガイド」の AWS Direct Connect「接続をモニタリングする CloudWatch アラームの作成」を参照してください。 AWS Direct Connect
Elastic Load Balancing - 受信したトラフィックを複数のアベイラビリティーゾーンの複数のターゲット (Amazon EC2 インスタンス、コンテナ、IP アドレスなど) に自動的に分散させます。	ロードバランサーイベント用に作成したアラームの通知を受信します。詳細については、「Classic Load Balancer のユーザーガイド」の「ロードバランサーの CloudWatch アラームを作成する」を参照してください。
Amazon Route 53 - ドメイン登録、DNS ルーティング、ヘルスチェックを提供します。	ヘルスチェックのステータスに不具合がある場合に通知を受信します。詳細については、『Amazon Route 53 デベロッパーガイド』の「ヘルスチェックのステータス (コンソール) に不具合がある場合に Amazon SNS 通知を受信するには 」を参照してください。

AWS のサービス	Amazon SNS を使用する利点
Amazon Virtual Private Cloud (Amazon VPC) - 定義した仮想ネットワーク内で AWS リソースを起動できます。	インターフェイスエンドポイントで発生する特定のイベントに関する通知を受信します。詳細については、『Amazon VPC User Guide』の「エンドポイントサービスの通知を作成および管理する」を参照してください。

セキュリティ、アイデンティティ、コンプライアンスサービス

AWS のサービス	Amazon SNS を使用する利点
AWS Directory Service - Microsoft Active Directory (AD) を AWS の他のサービスと併用するための複数の方法を提供します。	ディレクトリのステータスが変更されたときにメールまたはテキストメッセージ (SMS) を受け取ります。詳細については、『AWS Directory Service 管理ガイド』の「ディレクトリステータス通知の構成」を参照してください。
Amazon GuardDuty — AWS環境内の予期しないアクティビティや、不正または悪意のあるアクティビティを特定するのに役立つ継続的なセキュリティモニタリングを提供します。	新しくリリースされた結果タイプ、既存の結果タイプに関する更新などの機能の変更に関する通知を受信します。詳細については、「Amazon GuardDuty ユーザーガイド」の GuardDuty 「お知らせ SNS トピックにサブスクライブする」を参照してください。
Amazon Inspector - Amazon EC2 インスタンスのネットワークアクセシビリティとそれらのインスタンスで実行されるアプリケーションのセキュリティ状態をテストします。	Amazon Inspector イベントの通知を受信します。詳細については、『Amazon Inspector ユーザーガイド』の「Amazon Inspector 通知用の SNS トピックのセットアップ」を参照してください。
AWS Security Hub — AWS のセキュリティチェックを自動化し、セキュリティアラートを一元化します。	AWS Security Hub のコントロールや標準の追加、編集、または廃止に関する通知など、AWS Security Hub 発表に関する通知を受け取ります。詳細については、「Amazon

AWS のサービス	Amazon SNS を使用する利点
	Simple Notification Service を使用した AWS Security Hub の発表のサブスクライブ」を参照してください。

サーバーレスサービス

AWS のサービス	Amazon SNS を使用する利点
Amazon DynamoDB - フルマネージドの NoSQL データベースサービスでシームレスなスケーラビリティの高速で予測可能なパフォーマンスを提供します。	メンテナンスイベントが発生したときに通知を受信します。詳細については、『Amazon DynamoDB デベロッパーガイド』の「DAX クラスターの設定をカスタマイズする」を参照してください。
Amazon EventBridge – 独自のアプリケーション、 software-as-a-service (SaaS) アプリケーション、および AWSのサービスからリアルタイムデータのストリームを配信し、そのデータを Amazon SNS などのターゲットにルーティングします。以前は CloudWatch Events と呼ばれ EventBridge ていました。	EventBridge イベントの通知を受信します。詳細については、「Amazon EventBridge ユーザーガイド」の「Amazon ターゲット EventBridge 」を参照してください。
AWS Lambda - サーバーをプロビジョニングまたは管理しなくてもコードを実行できます。	SNS トピックを Lambda デッドレターキューまたは Lambda の送信先として設定することで、関数出力データを受信します。詳細については、『AWS Lambda デベロッパーガイド』の「非同期呼び出し」を参照してください。

ストレージサービス

AWS のサービス	Amazon SNS を使用する利点
AWS Backup - クラウド内およびオンプレミスの AWS のサービス間でデータのバックアップを一元化および自動化するのに役立ちます。	AWS Backup イベントの通知を受信します。詳細については、『AWS Backup デベロッパーガイド』の「Amazon SNS を使用して AWS Backup イベントを追跡する」を参照してください。
Amazon Elastic File System - Amazon EC2 インスタンスのファイルストレージを提供します。	Amazon EFS イベント用に作成したアラームの通知を受信します。詳細については、『Amazon Elastic File System User Guide』の「自動モニタリングツール」を参照してください。
Amazon S3 Glacier - 使用頻度の低いデータ用のストレージを提供します。	ボールトの通知設定で、ジョブが完了するとメッセージが SNS トピックに送信されるように設定します。詳細については、『Amazon S3 Glacier デベロッパーガイド』の「Amazon S3 Glacier でのボールト通知を設定する」を参照してください。
Amazon Simple Storage Service (Amazon S3) - オブジェクトストレージを提供します。	オブジェクトがレプリケート先リージョンにレプリケートされない場合、Amazon S3 バケットまたはまれなインスタンスに変更が生じると通知を受信します。詳細については、『Amazon Simple Storage Service ユーザーガイド』の「チュートリアル: 通知 (SNS トピックまたは SQS キュー) 用のバケットを設定する」および「レプリケーションメトリクスと Amazon S3 イベント通知による進捗状況のモニタリング」を参照してください。
AWS Snowball — 物理ストレージデバイスを使用して、Amazon S3 とオンサイトのデータ	Snowball ジョブの通知を受信します。詳細については、次を参照してください。 •

AWS のサービス	Amazon SNS を使用する利点
ストレージロケーション間で大量のデータを faster-than-internet 高速に転送します。	『AWS Snowball ユーザーガイド』の「Snowball 通知」 • ステップ 5: 「AWS Snowball Edge デベロッパーガイド」の通知設定を選択します • ステップ 5: 「AWS Snowball ユーザーガイド」の通知設定を選択します

追加のイベントソース

ソース	Amazon SNS を使用する利点
AWS 毎日の機能更新	Amazon SNS トピックを通じて、AWS のリリースや更新に関する詳細情報をタイムリーに受け取ることができます。これらのリリースには、AWS リージョン、AWS のサービス、Amazon VPC エンドポイント、AWSService Quotas とAWS のサービス統合された Amazon EC2 インスタンスタイプ、Amazon SageMaker インスタンスタイプ、Amazon Nimble Studio インスタンスタイプ、Amazon RDS データベースエンジンバージョン、および Amazon MSK Apache Kafka バージョンが含まれます。詳細については、AWS ニュースブログの「Amazon SNS 経由で AWS の毎日の機能更新をサブスクライブする」を参照してください。
AWS IP アドレスの範囲	Amazon SNS トピックを通じて AWS IP 範囲への変更の通知を受け取ります。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS IP アドレス範囲の通知」と、「AWS ニュースブログ」の「Amazon SNS 経

ソース	Amazon SNS を使用する利点
	由で AWS パブリック IP アドレスの変更をサブスクライブする」を参照してください。

イベント駆動型のコンピューティングの詳細については、次のソースを参照してください。

- イベント駆動型のアーキテクチャとは
- Amazon SNS と AWS コンピューティングを使用したイベント駆動型コンピューティング、ストレージ、データベース、AWS コンピューティングブログのネットワーキングサービス
- AWS コンピューティングブログで AWS イベントフォークパイプラインを使用したでイベント駆動型のアーキテクチャを強化する

Amazon SNS イベントの送信先

このページには、イベントに関する情報を受信できるすべての送信先が、 application-to-application (A2A) メッセージングおよび application-to-person (A2P) 通知ごとにグループ化されて一覧表示されます。

> **Note**
>
> Amazon SNS は 2020 年 10 月、FIFO トピックを導入しました。現在、ほとんどの AWS サービスは、SNS 標準トピックからのイベントの受信のみをサポートしています。Amazon SQS は、SNS 標準トピックと FIFO トピックの両方からのイベントの受信をサポートしています。

A2A 送信先

イベントの発行先	Amazon SNS を使用する利点
Amazon Data Firehose	アーカイブおよび分析の目的で、イベントを配信ストリームに配信します。配信ストリームを通じて、Amazon Simple Storage Service (Amazon S3)、Amazon Redshift、Amazon OpenSearch Service (OpenSearch Service)

イベントの発行先	Amazon SNS を使用する利点
	などのAWS送信先、または Datadog、New Relic、MongoDB、Splunk などのサードパーティーの送信先にイベントを配信できます。詳細については、「Firehose 配信ストリームへのファンアウト」を参照してください。
AWS Lambda	カスタムビジネスロジックの実行をトリガーする関数にイベントを配信します。詳細については、「Lambda 関数へのファンアウト」を参照してください。
Amazon SQS	アプリケーション統合の目的でイベントをキューに配信します。詳細については、「Amazon SQS キューへのファンアウト」を参照してください。
AWS Event Fork Pipelines	イベントのバックアップとストレージ、イベントの検索と分析、またはイベント再生パイプラインにイベントを配信します。詳細については、「AWS Event Fork Pipelines へのファンアウト」を参照してください。
HTTP/S	外部 webhook にイベントを配信します。詳細については、「HTTP(S) エンドポイントへのファンアウト」を参照してください。

A2Pの送信先

イベントの発行先	Amazon SNS を使用する利点
SMS	イベントをテキストメッセージとして携帯電話に配信します。詳細については、「モバイルテキストメッセージング (SMS)」を参照してください。

イベントの発行先	Amazon SNS を使用する利点
Email(メール)	イベントを E メールメッセージとして受信トレイに配信します。詳細については、「E メール通知」を参照してください。
プラットフォームエンドポイント	ネイティブプッシュ通知として携帯電話にイベントを配信します。詳細については、「モバイルプッシュ通知」を参照してください。
AWS Chatbot	Amazon Chime チャットルームや Slack チャンネルにイベントを配信します。詳細については、次のページの「AWS Chatbot 管理者ガイド」を参照してください。 • Amazon Chime AWS Chatbotでのセットアップ • Slack AWS Chatbotを使用したセットアップ • AWS Chatbot を他の AWS サービスと併用する
PagerDuty	オンコールチームにオペレーション上のインサイトを配信します。詳細については、「AWS管理とガバナンスブログ」の「Amazon DevOps Guru PagerDuty を使用して ML を活用した運用上のインサイトをオンコールチームに配信する」を参照してください。

> **Note**
>
> ネイティブ AWS イベントとカスタムイベントの両方をチャットアプリケーションに配信することができます。
>
> • ネイティブ AWS イベント — AWS Chatbot を使用してネイティブ AWS イベントを Amazon SNS トピック経由で Amazon Chime と Slack に送信できます。サポートされて

いるネイティブAWSイベントのセットには、AWS Billing and Cost Management、AWS Health、、AWS CloudFormationAmazon CloudWatchなどのイベントが含まれます。詳細については、『AWS Chatbot 管理者ガイド』の「Using AWS Chatbot with other services」を参照してください。

- カスタムイベント — Amazon SNS トピックを通じて、Amazon Chime、Slack、Microsoft Teams にカスタムイベントを送信することもできます。これを行うには、カスタムイベントを SNS トピックに発行し、サブスクライブされた Lambda 関数にイベントを配信します。Lambda 関数は、チャットアプリケーションの webhook を使用して、受信者にイベントを配信します。詳細については、「webhook を使用して Amazon SNS メッセージを Amazon Chime、Slack、または Microsoft Teams に送信するには」を参照してください。

Amazon SNS のアクセスをセットアップする

Amazon SNS を初めて使用する前に、以下のステップを完了する必要があります。

トピック

ステップ 1: AWS アカウント と IAM ユーザーを作成する

AWS のサービスにアクセスするには、まず を作成する必要がありますAWS アカウント。を使用して AWS アカウント 、アクティビティレポートと使用状況レポートを表示し、認証とアクセスを管理できます。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次のステップを実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

   にサインアップすると AWS アカウント、 AWS アカウント ルートユーザーが作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] を選んで、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理できます。

管理アクセス権を持つユーザーを作成する

にサインアップしたら AWS アカウント、 AWS アカウント ルートユーザーを保護し、 を有効にして AWS IAM Identity Center、日常的なタスクにルートユーザーを使用しないように管理ユーザーを作成します。

AWS アカウント ルートユーザーの保護

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、「IAM ユーザーガイド」の AWS アカウント 「ルートユーザーの仮想 MFA デバイスを有効にする (コンソール) 」を参照してください。

管理アクセス権を持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。
2. IAM アイデンティティセンターで、ユーザーに管理アクセス権を付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 ユーザーガイド」の「デフォルト でユーザーアクセスを設定する IAM アイデンティティセンターディレクトリAWS IAM Identity Center 」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、「 AWS サインイン ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するベストプラクティスに従うアクセス許可セットを作成します。

   「AWS IAM Identity Center ユーザーガイド」の「Create a permission set」の手順に従ってください。
2. ユーザーをグループに割り当ててから、そのグループにシングルサインオンアクセスを割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「Add groups」を参照してください。

次のステップ

Amazon SNS を使用する準備が整ったら、トピックの作成、トピックのサブスクリプションの作成、トピックへのメッセージの発行、およびサブスクリプションとトピックの削除を行うことで、使用を開始します。

Amazon SNS の開始方法

このセクションでは、Amazon SNS の理解を役立つように Amazon SNS コンソールを使用してトピック、サブスクリプション、およびメッセージを管理する方法について説明します。

トピック

前提条件

開始する前に、「Amazon SNS のアクセスをセットアップする」のステップを完了します。

ステップ 1: トピックを作成する

1. Amazon SNS コンソールにサインインします。
2. 左のナビゲーションペインで、[トピック] を選択します。
3. [トピック] ページで、[トピックの作成] を選択します。
4. デフォルトでは、コンソールは FIFO トピックを作成します。[Standard] を選択します。
5. 詳細セクションに、 などのトピックの名前を入力します*MyTopic*。
6. フォームの最下部までスクロールし、[トピックの作成] を選択します。

   コンソールに新しいトピックの [詳細] ページが表示されます。

ステップ 2: トピックに対するサブスクリプションを作成する

1. 左のナビゲーションペインで、[サブスクリプション] を選択します。
2. [サブスクリプション] ページで [サブスクリプションの作成] を選択します。

3. [サブスクリプションの作成] ページで、[トピック ARN] フィールドを選択し、AWS アカウントのトピックを一覧表示します。
4. 上記のステップで作成したトピックを選択します。
5. [プロトコル] で [E メール] を選択します。
6. [エンドポイント] に、通知を受信するために使用できる E メールアドレスを入力します。
7. [サブスクリプションの作成] を選択します。

   コンソールが新しいサブスクリプションの [詳細] ページを開きます。
8. E メールの受信トレイを確認し、AWS 通知の E メール内の [サブスクリプションの確認] を選択します。通常、送信者 ID は「no-reply@sns.amazonaws.com」です。
9. Amazon SNS がウェブブラウザを開き、サブスクリプション ID とともにサブスクリプションの確認を表示します。

ステップ 3: トピックにメッセージを発行する

1. 左のナビゲーションペインで、[トピック] を選択します。
2. [トピック] ページで、前に作成したトピックを選択し、[メッセージの発行] を選択します。

   コンソールが [トピックにメッセージを発行] ページを開きます。
3. (オプション) [メッセージの詳細] セクションで、以下のような [件名] を入力します。

   ```
   Hello from Amazon SNS!
   ```
4. [メッセージ本文] セクションで、[すべての配信プロトコルに同一のペイロード。] をクリックし、以下のようなメッセージ本文を入力します。

   ```
   Publishing a message to an SNS topic.
   ```
5. [メッセージの発行] を選択します。

   メッセージはトピックに発行され、コンソールがトピックの [詳細] ページを開きます。
6. E メールの受信トレイを確認し、発行されたメッセージを含む Amazon SNS からの E メールを受信したことを確認します。

ステップ 4: サブスクリプションとトピックを削除する

1. ナビゲーションパネルで、[サブスクリプション] を選択します。
2. [サブスクリプション] グループの設定ページで、確認済みのサブスクリプションを選択し、次に [削除] を選択します。

   > **Note**
   >
   > 保留中の確認は削除できません。48 時間後、Amazon SNS は自動的に削除します。

3. [サブスクリプションの削除] ダイアログボックスで、[削除] を選択します。

   サブスクリプションが削除されます。
4. ナビゲーションパネルで、[トピック] を選択します。
5. [トピック] ページで、トピックを選択し、次に [削除] を選択します。

   > **Important**
   >
   > トピックを削除すると、そのトピックのサブスクリプションもすべて削除されます。

6. トピックの削除*MyTopic*ダイアログボックスで、`delete me`「」と入力し、「削除」を選択します。

   トピックが削除されます。

次のステップ

これでサブスクリプションを使用してトピックを作成し、トピックにメッセージを送信したので、以下を試すことができます。

- [AWS デベロッパーセンター] の詳細。
- [セキュリティ] セクションで、データのアクセスの保護について説明します。
- トピックのサーバー側の暗号化を有効にします。
- 暗号化された Amazon Simple Queue Service (Amazon SQS) キューがサブスクライブされ、トピックのサーバー側の暗号化を有効にします。
- AWS Event Fork Pipelines をトピックにサブスクライブします。

Amazon SNS を設定する

Amazon SNS コンソールを使用して、Amazon SNS トピックとサブスクリプションを作成および設定します。Amazon SNS の詳細については、「Amazon SNS とは」を参照してください。

トピック

- Amazon SNS トピックを作成する
- Amazon SNS トピックへサブスクライブする
- Amazon SNS トピックとサブスクリプションの削除
- Amazon SNS トピックのタグ付け

Amazon SNS トピックを作成する

Amazon SNS トピックは、通信チャネルとして機能する論理アクセスポイントです。トピックでは、複数のエンドポイント (Amazon SQS、HTTP/S AWS Lambda、E メールアドレスなど) をグループ化できます。

メッセージプロデューサーシステム (e コマースウェブサイトなど) で当システムのメッセージを必要とする他の複数のサービス (チェックアウトシステムやフルフィルメントシステムなど) にメッセージをブロードキャストするには、プロデューサーシステムのトピックを作成できます。

Amazon SNS で最も一般的な最初のタスクは、トピックの作成です。このページでは AWS Management Console、、、を使用してトピックを作成する方法を説明します。 AWS SDK for Java AWS SDK for .NET

作成時に、トピックタイプ (標準または FIFO) を選択し、トピックに名前を付けます。トピックを作成したら、トピックのタイプや名前を変更することはできません。その他の設定項目はすべて、トピックの作成時にオプションであり、後で編集できます。

> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はトピック名に追加しないでください。トピック名には、 CloudWatch ログを含む他のAmazon Web Services からアクセスできます。トピック名は、プライベートデータや機密データとして使用することを意図していません。

トピック

- を使用してトピックを作成するには AWS Management Console
- SDK を使用してトピックを作成するには AWS

を使用してトピックを作成するには AWS Management Console

1. Amazon SNS コンソールにサインインします。
2. 次のいずれかを行います。
   - これまでにトピックを作成したことがない場合は、ホームページの Amazon SNS の説明をお読みください。 AWS アカウント
   - AWS アカウント 以前にトピックを作成したことがある場合は、ナビゲーションパネルで [Topics] を選択します。
3. [トピック] ページで、[トピックの作成] を選択します。
4. [サブスクリプションの作成] ページで [詳細] セクションで、以下を実行します。
   a. [タイプ] で、トピックタイプ (標準またはFIFO) を選択します。
   b. トピックの名前を入力します。FIFO トピックで、名前の末尾に .fifo を追加します。
   c. (オプション) トピックの表示名を入力します。

      > **Important**
      >
      > E メールエンドポイントをサブスクライブする場合、Amazon SNS トピックの表示名と送信 E メールアドレス (例えば、no-reply@sns.amazonaws.com) の合計文字数が 320 UTF-8 文字を超えてはなりません。Amazon SNS トピックの表示名を設定する前に、サードパーティーのエンコードツールを使用して送信アドレスの長さを確認できます。

   d. (オプション) FIFO トピックで、[コンテンツベースのメッセージ重複排除] を選択して、デフォルトのメッセージの重複排除を有効にします。詳細については、「FIFO トピックのメッセージ重複除外」を参照してください。
5. (オプション) [暗号化] セクションを展開し、以下の操作を実行します。詳細については、「保管中の暗号化」を参照してください。
   a. [暗号化の有効化] を選択します。

b. AWS KMS キーを指定します。詳細については、「重要な用語」を参照してください。

KMS タイプごとに、[Description] (説明)、[Account] (アカウント)、および [KMS ARN] が表示されます。

> **Important**
>
> KMS の所有者ではない場合、または `kms:ListAliases` および `kms:DescribeKey` の許可がないアカウントでログインした場合、Amazon SNS コンソールで KMS に関する情報を表示できません。
> これらの許可を付与するように、KMS の所有者へ依頼してください。詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMS API アクセス権限: アクションとリソースのリファレンス」を参照してください。

- Amazon SNS (デフォルト) エイリアス/AWS/SNS AWS 用のマネージド KMS がデフォルトで選択されています。

  > **Note**
  >
  > 以下に留意してください。
  >
  > - 初めてを使用してトピックの AWS Management Console Amazon SNS AWS 用のマネージド KMS AWS KMS を指定すると、Amazon SNS AWS 用のマネージド KMS が作成されます。
  > - または、SSE `Publish` が有効になっているトピックでアクションを初めて使用したときに、Amazon SNS AWS 用のマネージド KMS AWS KMS を作成します。

- AWS アカウントからカスタム KMS を使用するには、KMS キーフィールドを選択し、リストからカスタム KMS を選択します。

  > **Note**
  >
  > カスタム KMS の作成手順については、AWS Key Management Service デベロッパーガイドの「キーの作成」を参照してください。

- AWS AWS 自分のアカウントまたは別のアカウントのカスタム KMS ARN を使用するには、KMS キーフィールドに入力します。

6. (オプション) デフォルトでは、トピックの所有者のみがトピックを発行またはサブスクライブできます。追加のアクセス許可を設定するには、[アクセスポリシー] セクションを展開します。詳細については、「Amazon SNS での Identity and Access Management」および「Amazon SNS アクセスコントロールのケース例」を参照してください。

   > **Note**
   >
   > コンソールを使用してトピックを作成すると、デフォルトのポリシーでは `aws:SourceOwner` 条件キーが使用されます。このキーは `aws:SourceAccount` に類似しています。

7. (オプション) 失敗したメッセージ配信試行を Amazon SNS で再試行する方法を設定するには、[配信再試行ポリシー (HTTP/S)] セクションを展開します。詳細については、「Amazon SNS メッセージ配信の再試行」を参照してください。
8. (オプション) Amazon SNS がメッセージの配信を記録する方法を設定するには CloudWatch、[配信ステータスロギング] セクションを展開します。詳細については、「Amazon SNS メッセージ配信ステータス」を参照してください。
9. (オプション) トピックにメタデータタグを追加するには、[タグ] セクションを展開し、[キー] と [値] (オプション) に入力し、[タグの追加] を選択します。詳細については、「Amazon SNS トピックのタグ付け」を参照してください。
10. [Create topic] (トピックの作成) を選択します。

    トピックが作成され、***MyTopic***ページが表示されます。

    トピックの名前、ARN、(オプション) 表示名、 AWS トピック所有者のアカウント ID が [詳細] セクションに表示されます。
11. トピック ARN をクリップボードにコピーします。例:

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

SDK を使用してトピックを作成するには AWS

AWS SDK を使用するには、認証情報を使用して設定する必要があります。詳細については、「AWS SDK とツールのリファレンスガイド」の「共有設定ファイルと認証情報ファイル」を参照してください。

次のコード例は使用方法を示していますCreateTopic。

.NET

AWS SDK for .NET

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックを作成して、個別の名前を付けます。

```
    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;
        }
    }
```

名前と特定の FIFO および重複除外属性を使用して新しいトピックを作成します。

```
    /// <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;
    }
```

- API の詳細については、AWS SDK for .NET API CreateTopicリファレンスのを参照してください。

C++

SDK for C++

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、AWS SDK for C++ API CreateTopicリファレンスのを参照してください。

CLI

AWS CLI

SNS トピックを作成するには

次の create-topic の例では、my-topic という名前の SNS トピックを作成します。

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

出力:

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

詳細については、『AWSコマンドラインインターフェイスユーザーガイド』の「Amazon SQS と Amazon SNS AWS でのコマンドラインインターフェイスの使用」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」CreateTopicのを参照してください。

Go

SDK for Go V2

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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
}
```

- API の詳細については、AWS SDK for Go API CreateTopicリファレンスのを参照してください。

Java

SDK for Java 2.x

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API CreateTopicリファレンスのを参照してください。

JavaScript

JavaScript (v3) 用の SDK

> **Note**
>
> にはまだまだあります。 GitHub用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、AWS SDK for JavaScript API CreateTopicリファレンスのを参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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()
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」のを参照してくださいCreateTopic。

PHP

SDK for PHP

> **Note**
>
> にはまだまだあります。 GitHub用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API CreateTopicリファレンスのを参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、『AWS SDK for Python (Boto3) API リファレンス』のを参照してくださいCreateTopic。

Ruby

SDK for Ruby

> **Note**
>
> にはまだまだあります。 GitHub用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for Ruby API CreateTopicリファレンスのを参照してください。

Rust

SDK for Rust

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」のを参照してくださいCreateTopic。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスのを参照してくださいCreateTopic。

Amazon SNS トピックへサブスクライブする

トピックに対して発行されたメッセージを受信するには、そのトピックへのエンドポイントをサブスクライブする必要があります。エンドポイントをトピックにサブスクライブしてサブスクリプションが確認されると、エンドポイントでは、関連付けられたトピックに発行されたメッセージの受信を開始できます。

> **Note**
>
> HTTP (S) エンドポイント、E メールアドレス、その他の AWS アカウント の AWS リソースは、メッセージを受信する前にサブスクリプションの確認が要求です。

エンドポイントを Amazon SNS トピックにサブスクライブするには

1. Amazon SNS コンソールにサインインします。
2. 左のナビゲーションペインで、[サブスクリプション] を選択します。
3. [サブスクリプション] ページで [サブスクリプションの作成] を選択します。
4. [サブスクリプションの作成] ページで [詳細] セクションで、以下を実行します。

   a. [トピック ARN] は、トピックの Amazon リソースネーム (ARN) を選択します。この値は、`arn:aws:sns:us-east-2:123456789012:your_topic` などの Amazon SNS トピックを作成したときに生成された AWS ARNです。

   b. [プロトコル] でエンドポイントタイプを選択します。使用可能なエンドポイントタイプは次のとおりです。

      - HTTP/HTTPS
      - Email/Email-JSON
      - Amazon Data Firehose

- Amazon SQS

> **Note**
>
> SNS FIFO トピックにサブスクライブするには、このオプションを選択します。

- AWS Lambda
- プラットフォームアプリケーションエンドポイント
- SMS

c. [エンドポイント] に、エンドポイント値 (E メールアドレスや Amazon SQS キューの ARN など) を入力します。

d. Firehose エンドポイントのみ: サブスクリプションロール ARN には、Firehose 配信ストリームへの書き込み用に作成した IAM ロールの ARN を指定します。詳細については、「Firehose 配信ストリームを Amazon SNS トピックにサブスクライブするための前提条件」を参照してください。

e. (オプション) Firehose、Amazon SQS、HTTP/S エンドポイントでは、raw メッセージ配信を有効にすることもできます。詳細については、「Amazon SNS raw メッセージの配信」を参照してください。

f. (オプション) フィルターポリシーを設定するには、[サブスクリプションのフィルターポリシー] セクションを展開します。詳細については、「Amazon SNS サブスクリプションフィルターポリシー」を参照してください。

g. (オプション) ペイロードベースのフィルタリングを有効にするには、`Filter Policy Scope` を `MessageBody` に設定します。詳細については、「Amazon SNS サブスクリプションフィルターポリシーの範囲」を参照してください。

h. (オプション) サブスクリプションのデッドレターキューを設定するには、Redrive ポリシー (デッドレターキュー)を展開します。詳細については、「Amazon SNS デッドレターキュー (DLQ)」を参照してください。

i. [サブスクリプションの作成] を選択します。

コンソールがサブスクリプションを作成し、サブスクリプションの [詳細] ページを開きます。

Amazon SNS トピックとサブスクリプションの削除

トピックが削除されると、関連するサブスクリプションは非同期的に削除されます。顧客はこれらのサブスクリプションにアクセスできますが、同じ名前を使用してトピックを再作成した場合でも、サブスクリプションはトピックに関連付けられなくなります。

サブスクライバーが削除されたトピックにメッセージを発行しようとすると、パブリッシャーはトピックが存在しないことを示すエラーメッセージを受け取ります。同様に、削除されたトピックをサブスクライブしようとすると、エラーメッセージが表示されます。

確認待ちのサブスクリプションは削除できません。Amazon SNS は、未確認のサブスクリプションを 48 時間後に自動的に削除します。

トピック

- を使用して Amazon SNS トピックまたはサブスクリプションを削除するには AWS Management Console
- AWS SDK を使用してサブスクリプションとトピックを削除するには

を使用して Amazon SNS トピックまたはサブスクリプションを削除するには AWS Management Console

を使用してトピックを削除するには AWS Management Console

1. Amazon SNS コンソールにサインインします。
2. 左のナビゲーションペインで、[トピック] を選択します。
3. [トピック] ページでトピックを選択し、[編集] を選択します。
4. [削除] ダイアログボックスに「`delete me`」と入力し、[削除] を選択します。

   コンソールがトピックを削除します。

を使用してサブスクリプションを削除するには AWS Management Console

1. Amazon SNS コンソールにサインインします。
2. 左のナビゲーションペインで、[サブスクリプション] を選択します。
3. サブスクリプションページで、ステータスが確認済み のサブスクリプションを選択し、削除を選択します。

4. [サブネットの削除] ダイアログボックスで、[削除] を選択します。

   コンソールはサブスクリプションを削除します。

AWS SDK を使用してサブスクリプションとトピックを削除するには

AWS SDK を使用するには、 認証情報を使用して設定する必要があります。詳細については、「AWS SDK とツールのリファレンスガイド」の「共有設定ファイルと認証情報ファイル」を参照してください。

以下のコード例は、`DeleteTopic` の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピック ARN でトピックを削除します。

```
/// <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;
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

SNS トピックを削除するには

次の delete-topic の例では、指定した SNS トピックを削除します。

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

このコマンドでは何も出力されません。

- API の詳細については、「 コマンドリファレンスDeleteTopic」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for Go

Java

SDK for Java 2.x

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> には他にもがあります GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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
  //   }
  // }
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、DeleteTopicAWS 「 SDK for Kotlin API リファレンス」の「」を参照してください。

PHP

SDK for PHP

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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());
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。AWS SDK for PHP

Python

SDK for Python (Boto3)

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、DeleteTopicAWS 「 SDK for Python (Boto3) API リファレンス」の「」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.
```

- API の詳細については、DeleteTopicAWS 「 SDK for SAP ABAP API リファレンス」の「」を参照してください。

Amazon SNS トピックのタグ付け

Amazon SNS では、Amazon SNS トピックのタグ付けがサポートされています。これにより、トピックに関連するコストを追跡および管理し、AWS Identity and Access Management (IAM) ポリシーのセキュリティを強化し、数千のトピックを簡単に検索またはフィルタリングできます。タグ付けすることで、AWS Resource Groups を使用して Amazon SNS トピックを管理できます。Resource Groups の詳細については、「AWS Resource Groups ユーザーガイド」を参照してください。

トピック

- コスト割り当てのタグ付け
- アクセス制御のタグ付け
- リソース検索およびフィルタリングのタグ付け
- Amazon SNS トピックタグを設定

コスト割り当てのタグ付け

コスト割り当てのために Amazon SNS トピックを整理および識別するには、トピックの目的を識別するタグを追加することができます。これは特に、多くのトピックを持っている場合に便利です。コスト配分タグを使用してAWSの請求書を整理し、自分のコスト構造に反映できます。そのためには、サインアップして AWS アカウントの請求書にタグキーおよび値を含めます。詳細については、「AWS Billing and Cost Management User Guide」の「Setting Up a Monthly Cost Allocation Report」を参照してください。

たとえば、次のように Amazon SNS のトピックのコストセンターや目的を表すタグを追加することができます。

リソース	キー	値
トピック 1	コストセンター	43289
	アプリケーション	注文処理
トピック 2	コストセンター	43289
	アプリケーション	お支払い方法
トピック 3	コストセンター	76585
	アプリケーション	アーカイブ

このタグ付けスキームでは、関連するタスクを実行している 2 つのトピックを同じコストセンターにグループ化する一方で、関連性のないアクティビティには別のコスト割り当てタグでタグ付けすることができます。

アクセス制御のタグ付け

AWS Identity and Access Management は、タグに基づくリソースへのアクセスの制御をサポートしています。リソースにタグ付け後、IAM ポリシーの条件要素でリソース タグに関する情報を提供し、タグベースのアクセスを管理します。Amazon SNS console (Amazon SNS コンソール) または AWS SDK を使用してリソースにタグ付けする方法については、「Configuring tags」(タグを設定する) を参照してください。

IAM アイデンティティのアクセスを制限できます。たとえば、キー environment および値 production のタグを含むすべてのAmazon SNS トピックへの Publish および PublishBatch のアクセスを制限し、他のすべての Amazon SNS トピックへのアクセスを許可することができます。次の例では、ポリシーにより、production でタグ付けされたトピックにメッセージを公開する機能が制限され、development でタグ付けされた トピックにメッセージを公開できるようになります。詳細については、IAM ユーザーガイドの [Controlling Access Using Tags] (タグを使用したアクセス制御) を参照してください。

> **Note**
>
> Publish の IAM の許可を設定すると、Publish および PublishBatch の両方に権限が設定されます。

```
{
  "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"
      }
    }
  }]
}
```

リソース検索およびフィルタリングのタグ付け

AWS アカウントには数万の Amazon SNS トピックを持つことができます (詳細については、「Amazon SNS Quotas」を参照)。トピックにタグ付けすることで、トピックの検索やフィルタリングのプロセスを簡素化することができます。

たとえば、本番環境に関連するトピックが数百個ある場合があります。これらのトピックを手動で検索するのではなく、指定されたタグを持つすべてのトピックを照会することができます。

```
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());
    }
}
```

Amazon SNS トピックタグを設定

このページでは、、、 AWS SDK AWS Management Console、および AWS CLI を使用して Amazon SNS トピックのタグを設定する方法を説明します。

> **Important**
>
> 個人情報 (PII) などの機密情報や秘匿性の高い情報はタグに追加しないようにします。タグは、他の Amazon Web Services のサービス (請求など) からアクセスできます。タグは、プライベートデータや機密データに使用することを意図していません。

トピック

- を使用して Amazon SNS トピックのタグを一覧表示、追加、削除する AWS Management Console
- AWS SDK を使ったトピックへのタグの追加
- Amazon SNS API アクションによるタグの管理
- ABAC をサポートする API アクション

を使用して Amazon SNS トピックのタグを一覧表示、追加、削除する AWS Management Console

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. [トピック] ページで、トピックを選択して [編集] を選択します。
4. [タグ] セクションを展開します。

   トピックに追加されているタグが一覧表示されます。
5. トピックのタグを変更します。
   - タグを追加するには、[Add tag] (タグの追加) を選択し、[Key] (キー) と [Value] (値) (オプション) を入力します。
   - タグを削除するには、キー値ペアの横にある [タグの削除] を選択します。
6. [変更を保存]をクリックします。

AWS SDK を使ったトピックへのタグの追加

AWS SDK を使用するには、認証情報を使用して設定する必要があります。詳細については、「AWS SDK とツールのリファレンスガイド」の「共有設定ファイルと認証情報ファイル」を参照してください。

次のコード例は使用方法を示していますTagResource。

CLI

AWS CLI

トピックにタグを追加するには

次の tag-resource の例では、指定した Amazon SNS トピックにメタデータタグを追加します。

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」TagResourceのを参照してください。

Java

SDK for Java 2.x

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API TagResourceリファレンスのを参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」のを参照してくださいTagResource。

Amazon SNS API アクションによるタグの管理

Amazon SNS API を使ってタグを管理するには、以下の API アクションを使用します。

- ListTagsForResource
- TagResource
- UntagResource

ABAC をサポートする API アクション

以下に、属性ベースのアクセスコントロール (ABAC) をサポートする API アクションの一覧を示します。ABAC の詳細については、「ABAC とは何ですか?」を参照してください。 AWS『IAM ユーザーガイド』にあります。

- AddPermission
- ConfirmSubscription
- DeleteTopic
- GetDataProtectionPolicy
- GetSubscriptionAttributes
- GetTopicAttributes
- ListSubscriptionsByTopic
- ListTagsForResource
- Publish
- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource

メッセージの順序付けと重複除外 (FIFO トピック)

Amazon SNS FIFO (先入れ先出し) トピックと Amazon SQS FIFO キューを組み合わせて、厳密なメッセージの順序付けとメッセージの重複排除を提供できます。これらの各サービスの FIFO 機能は連携して、ほぼリアルタイムでデータの一貫性を必要とする分散アプリケーションを統合する、フルマネージドサービスとして機能します。Amazon SQS 標準キューを Amazon SNS FIFO トピックにサブスクライブすると、ベストエフォート型の順序付けと少なくとも 1 回の配信が可能になります。

トピック

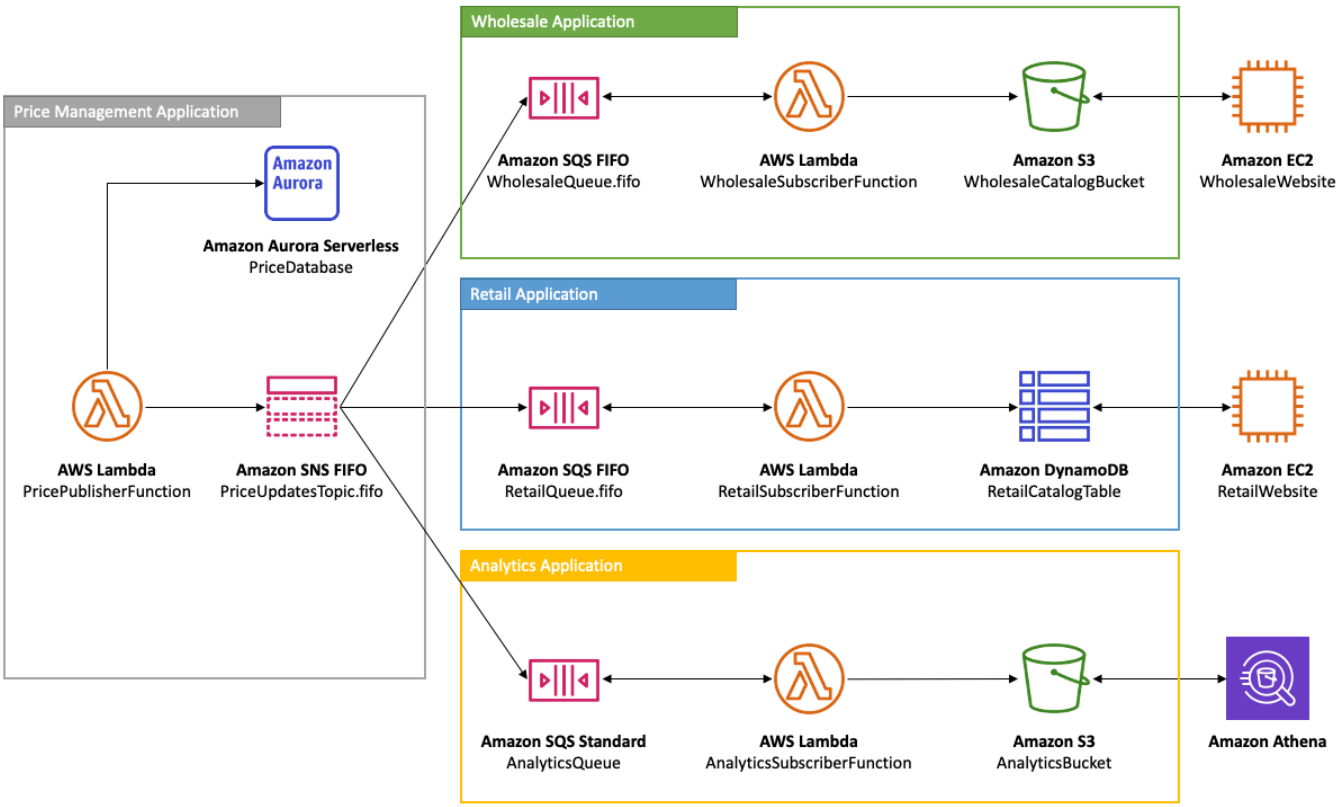
FIFO トピックのユースケース例

次の例は、自動車部品メーカーが Amazon SNS FIFO トピックと Amazon SQS キューを使用して構築した e コマースプラットフォームを示してします。プラットフォームは 4 つのサーバーレスアプリケーションで構成されています。

- インベントリ管理者は、価格管理アプリケーションを使用して、在庫の各品目の価格を設定します。この会社では、為替変動や市場需要、販売戦略の推移などにより、商品価格が変動する可能性があります。価格管理アプリケーションは、AWS Lambda 関数を使用して、価格が変更されるたびに Amazon SNS FIFO トピックに価格の更新を発行します。
- 卸売アプリケーションは、自動車修理工場や自動車メーカーがその自動車部品を一括で購入できるウェブサイトのバックエンドを提供します。価格変更通知を取得する場合、卸売アプリケーション

は Amazon SQS FIFO キューを価格管理アプリケーションの Amazon SNS FIFO トピックにサブスクライブします。

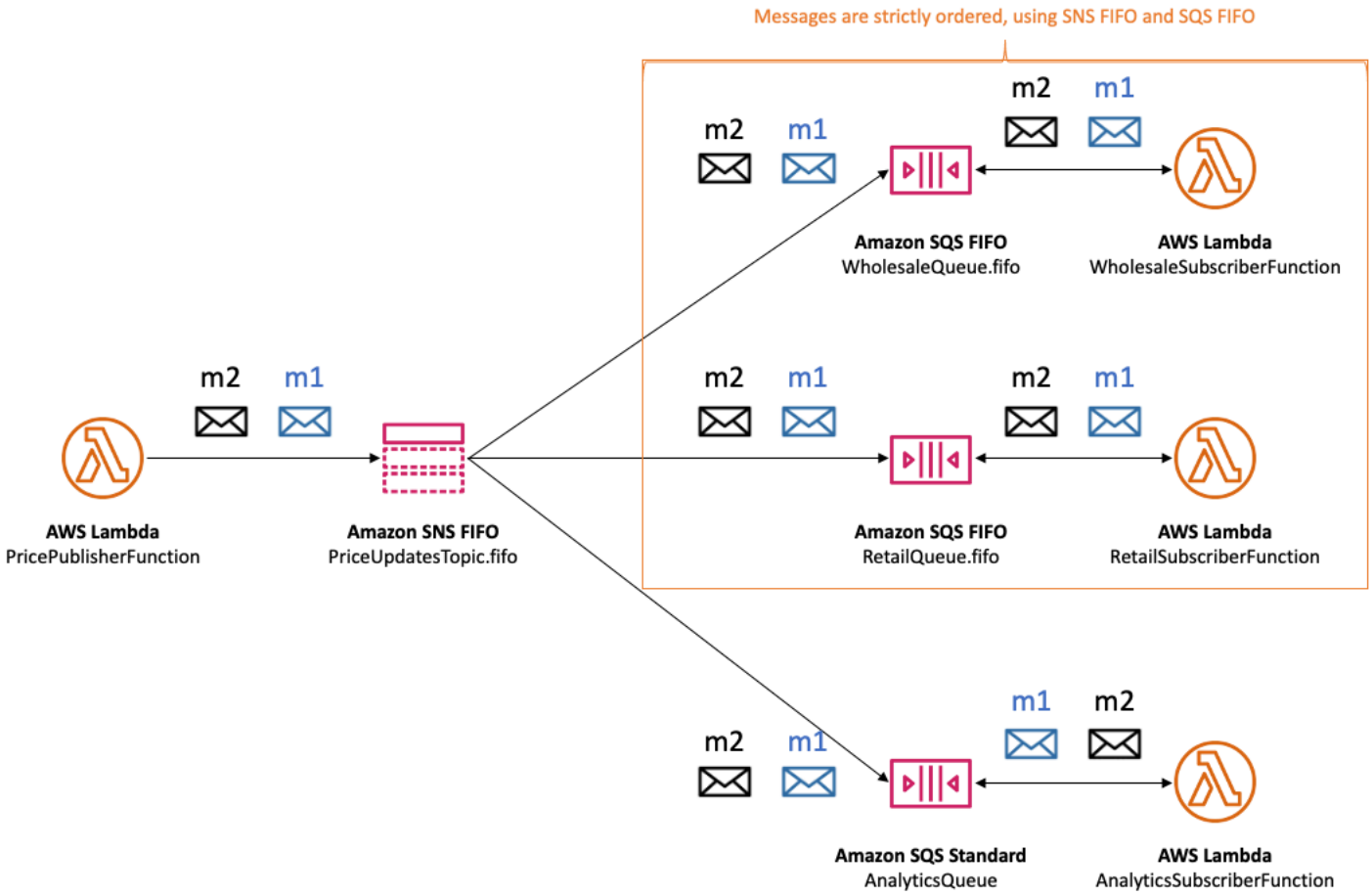
- 小売アプリケーションは、車の所有者や車のチューニング愛好家が自分の車の個々の自動車部品を購入できるウェブサイトのバックエンドを提供します。価格変更通知を取得する場合、小売アプリケーションも Amazon SQS FIFO キューを価格管理アプリケーションの Amazon SNS FIFO トピックにサブスクライブします。
- 分析アプリケーションは、価格の更新を集約して Amazon S3 バケットに保存し、Amazon Athena がビジネスインテリジェンス (BI) 目的でバケットをクエリできるようにします。価格変更通知を取得する場合、分析アプリケーションは Amazon SQS 標準キューを価格管理アプリケーションの Amazon SNS FIFO トピックにサブスクライブします。他のアプリケーションとは異なり、分析アプリケーションでは、価格更新を厳密に順序付ける必要はありません。



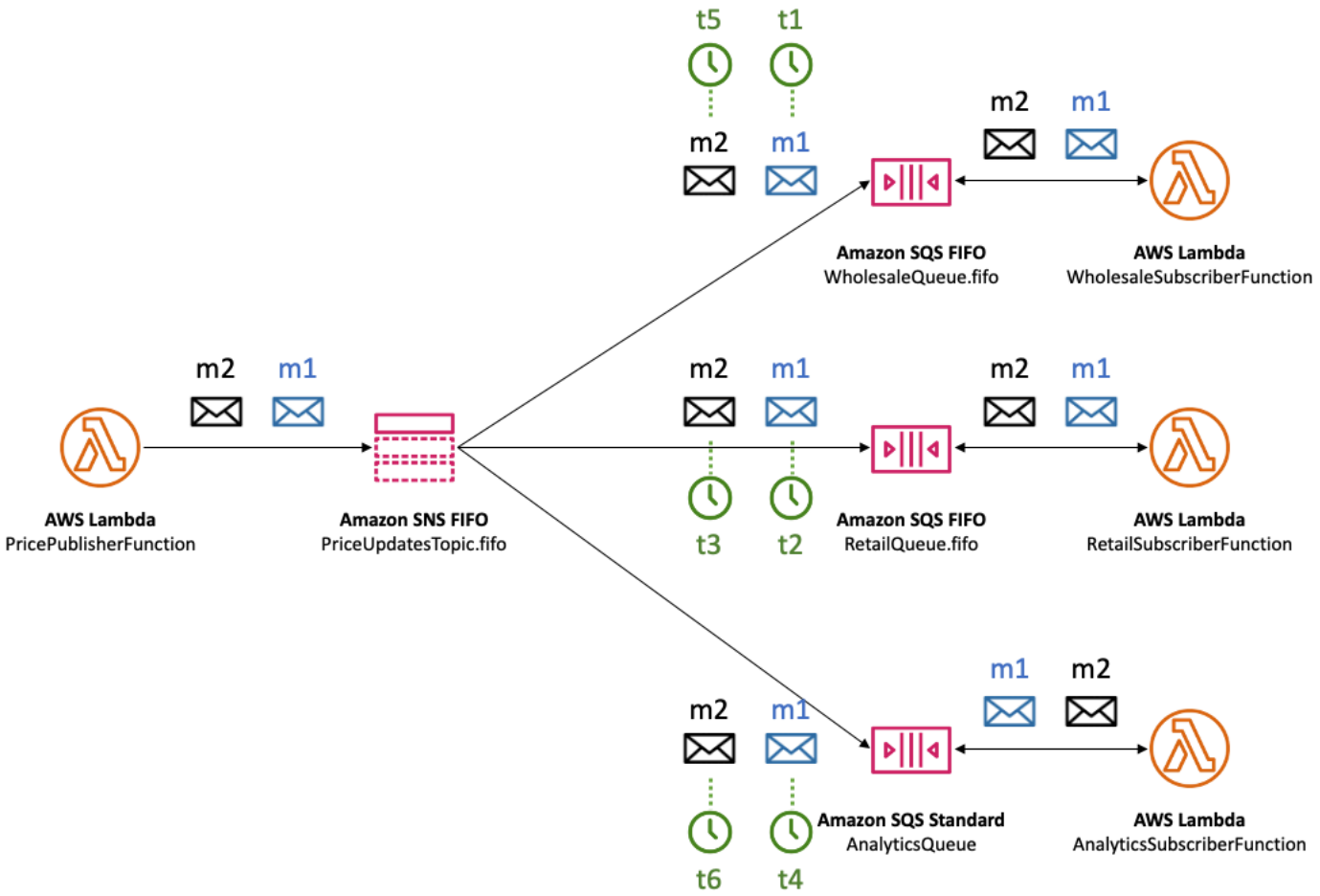
卸売アプリケーションおよび小売アプリケーションが正しい順序で価格の更新を受信するには、価格管理アプリケーションは、厳密に注文されたメッセージ配信システムを使用する必要があります。Amazon SNS FIFO トピックと Amazon SQS FIFO キューを使用すると、メッセージを順序どおりに重複なく処理できます。詳細については、「FIFO トピックのメッセージ順序の詳細」を参照してください。このユースケースを実装するコードスニペットについては、「FIFO トピックのコード例」を参照してください。

FIFO トピックのメッセージ順序の詳細

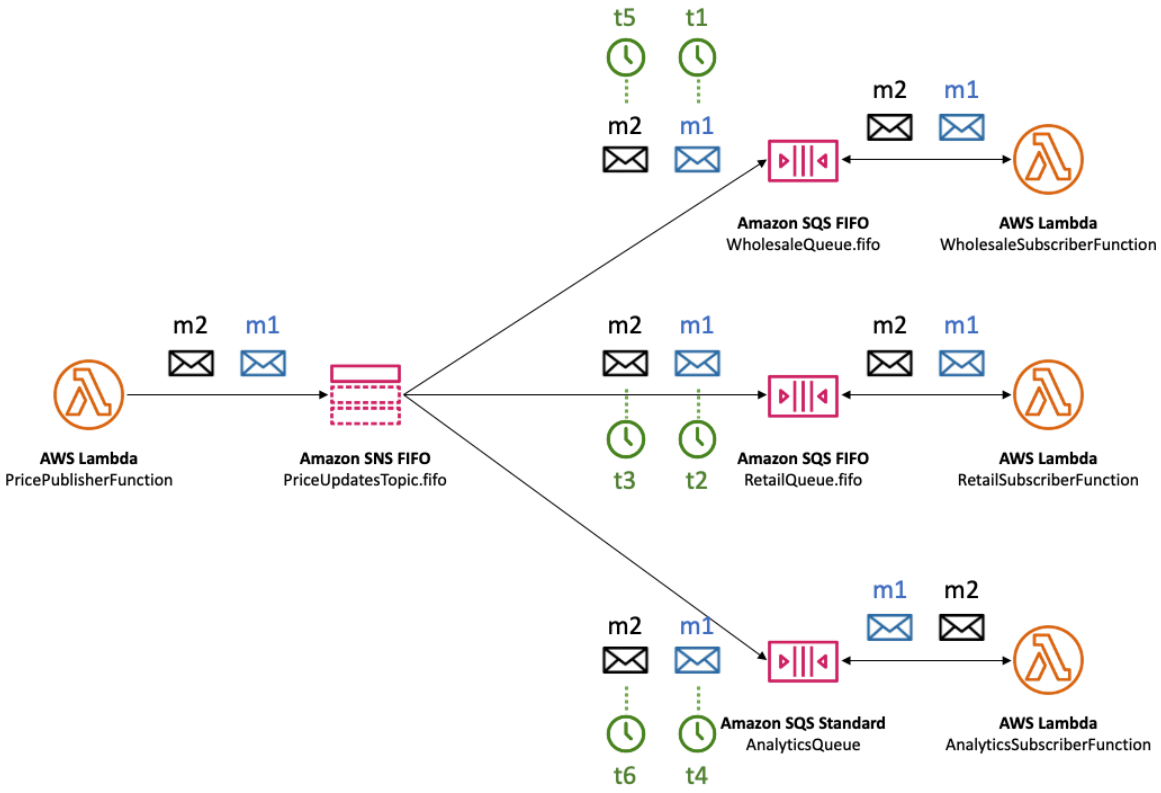
Amazon SNS FIFO トピックは、常に、メッセージがトピックに発行されるのとまったく同じ順序で、サブスクライブされた Amazon SQS FIFO キューにメッセージを 1 回だけ配信します。Amazon SQS FIFO キューをサブスクライブすると、キューのコンシューマーは、メッセージがキューに配信されるのとまったく同じ順序で、メッセージを重複することなく受信します。ただし、 SQS 標準キューをサブスクライブすると、キューのコンシューマーはメッセージを順序どおりではなく、さらに重複して受信する可能性があります。これにより、サブスクライバーをパブリッシャーからさらに切り離すことができ、次の図に示すように、「FIFO トピックのユースケース例」に基づいて、サブスクライバーはメッセージの使用とコストの最適化をより柔軟に行うことができます。



サブスクライバーの暗黙の順序付けはないことに注意してください。次の例は、メッセージ m1 が卸売サブスクライバー、小売サブスクライバー、分析サブスクライバーの順に配信されることを示しています。メッセージ m2 は、小売サブスクライバー、卸売サブスクライバー、分析サブスクライバーの順に配信されます。2 つのメッセージは異なる順序でサブスクライバーに配信されますが、メッセージの順序は Amazon SQS FIFO サブスクライバーごとに保持されます。各サブスクライバーは、他のサブスクライバーとは切り離して認識されます。

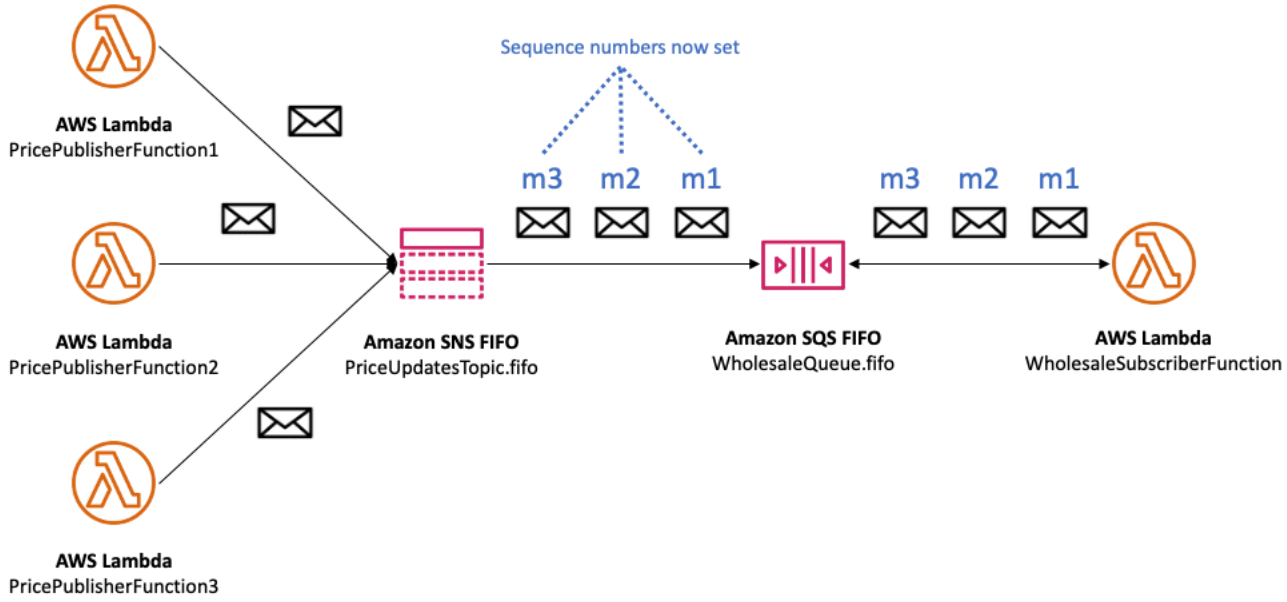


Amazon SQS キューサブスクライバーが到達不能になると、同期が失われる可能性があります。例えば、卸売アプリケーションキュー所有者が誤って Amazon SQS キューポリシーを変更し、Amazon SNS サービスプリンシパルがメッセージをキューに配信できなくなったとします。この場合、卸売キューへの価格更新配信は失敗し、小売キューと分析キューへの配信は成功するため、サブスクライバーは同期しなくなります。卸売アプリケーションキュー所有者がキューポリシーを修正すると、Amazon SNS はサブスクライブされたキューへのメッセージ配信を再開します。対応するサブスクリプションにデッドレターキューが設定されていない限り、誤設定されたキューを対象とするトピックに発行されたメッセージはすべてドロップされます。



複数のアプリケーション (または同じアプリケーション内の複数のスレッド) で、SNS FIFO トピックにメッセージを並行して発行できます。これを行うと、メッセージシーケンスを Amazon SNS サービスに効果的に委任します。確立されたメッセージのシーケンスを決定するには、シーケンス番号を確認します。

シーケンス番号は、Amazon SNS が各メッセージに割り当てる、連続しない大きな数字です。シーケンス番号の長さは 128 ビットで、メッセージグループごとに増え続けます。シーケンス番号は、メッセージ本文の一部として、サブスクライブされた Amazon SQS キューに渡されます。ただし、raw メッセージ配信を有効にすると、Amazon SQS キューに配信されるメッセージには、シーケンス番号やその他の Amazon SNS メッセージメタデータは含まれません。



Amazon SNS FIFO トピックは、メッセージグループのコンテキストで順序を定義します。詳細については、「FIFO トピックのメッセージのグループ化」を参照してください。

FIFO トピックのメッセージのグループ化

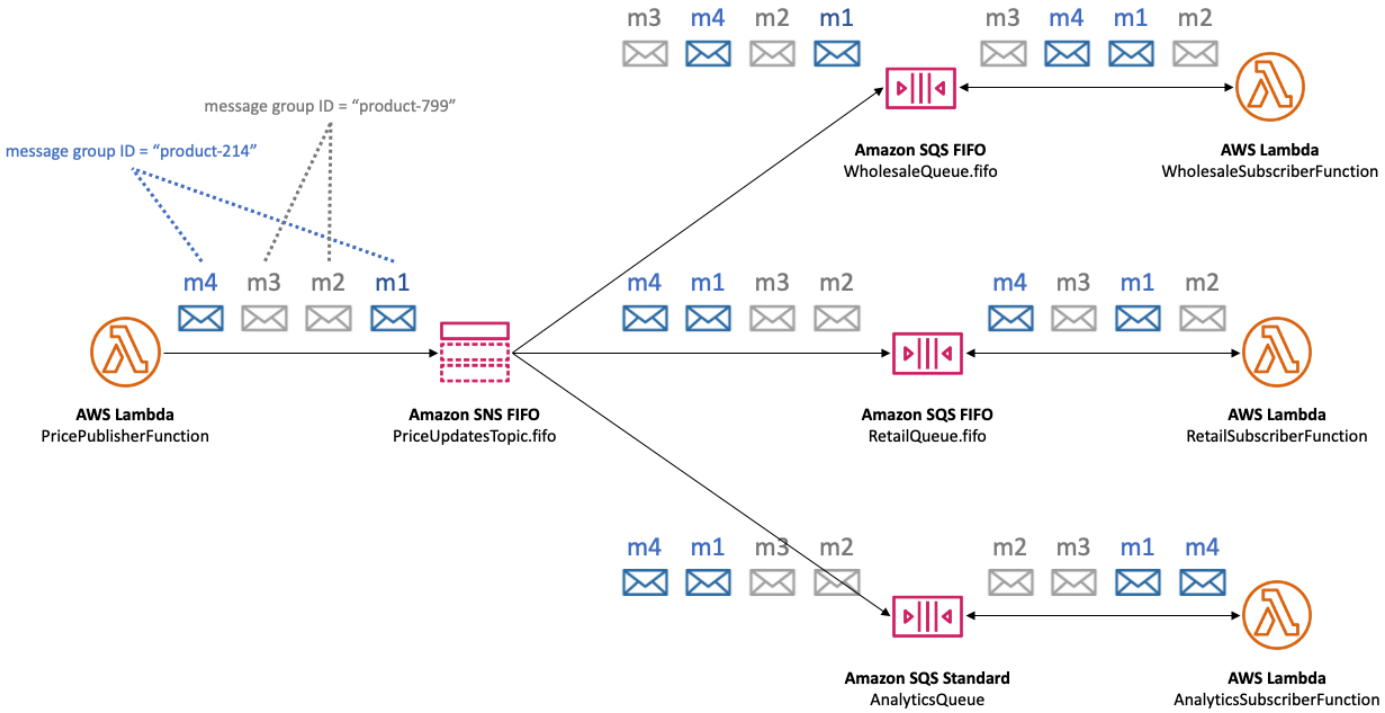
同じグループに属するメッセージは、グループに対する厳密な順序で 1 つずつ処理されます。

メッセージを Amazon SNS FIFO トピックに発行する際、メッセージのグループ ID を設定します。グループ ID は、メッセージが特定のメッセージグループに属することを指定する必須トークンです。SNS FIFO トピックは、グループ ID をサブスクライブされた Amazon SQS FIFO キューに渡します。SNS FIFO トピックまたは SQS FIFO キュー内のグループ ID の数に制限はありません。メッセージグループ ID は Amazon SQS 標準キューには渡されません。

メッセージグループとサブスクリプションの間にアフィニティはありません。したがって、任意のメッセージグループに発行されたメッセージは、サブスクリプションに添付されたフィルターポリシーに従って、すべてのサブスクライブ済みキューに配信されます。詳細については、FIFO トピックのメッセージ配信 および FIFO トピックのメッセージフィルター処理 を参照してください。

自動車部品価格管理のユースケース例では、プラットフォームで販売されている製品ごとに専用のメッセージグループがあります。すべての価格更新の処理には、同じ Amazon SNS FIFO トピックが使用されます。価格更新の順序は、単一の自動車部品製品のコンテキスト内で保持されますが、ない複数の製品にまたがっていません。この 仕組み を以下に示します。メッセージグループ ID が product-214 の製品の場合、メッセージ m1 はメッセージ m4 の前に処理されていることに注意してください。この順序は、Amazon SNS FIFO と Amazon SQS FIFO を使用するワークフロー全体を通

じて保持されます。同様に、メッセージグループ ID が product-799 の製品の場合、ワークフローが Amazon SNS FIFO と Amazon SQS FIFO を使用する限り、メッセージ m2 はメッセージ m3 の前に処理されます。ただし、Amazon SQS 標準キューを使用すると、メッセージの順序は保証されなくなり、メッセージグループは存在しません。product-214 および product-799 メッセージグループは互いに独立しているため、メッセージの順序付けには関係がありません。



メッセージグループ ID 別のデータ配布によりパフォーマンスを改善

配信スループットを最適化するために、Amazon SNS FIFO トピックは異なるメッセージグループからのメッセージを並列配信しますが、メッセージの順序は各メッセージグループ内で厳密に維持されます。個々のメッセージグループごとに、1 秒あたり最大 300 件のメッセージを配信できます。したがって、1 つのトピックで高いスループットを実現するには、多数の異なるメッセージグループ ID を使用してください。Amazon SNS FIFO トピックは、多様なメッセージグループのセットを利用することで、メッセージを多数の並列パーティションに自動的に配布します。

> **Note**
>
> Amazon SNS FIFO トピックは、グループの数に関係なく、メッセージグループ ID 全体にメッセージを均一に配布するように最適化されています。AWS では、パフォーマンスを最適化するために、個別のメッセージグループ ID を多数使用することをお勧めします。

スループットが高い Amazon SNS FIFO トピックに公開し、1 つ以上の Amazon SQS FIFO キューがサブスクライブされている場合は、キューで高スループットを有効にすることをお勧めします。詳細については、「Amazon Simple Queue Service デベロッパーガイド」の「FIFO キューの高スループット」を参照してください。

FIFO トピックのメッセージ配信

Amazon SNS FIFO (先入れ先出し) トピックは、Amazon SQS 標準キューと FIFO キューの両方への配信をサポートしており、ほぼリアルタイムのデータ整合性を必要とする分散アプリケーションを統合する際に柔軟性と制御をお客様に提供します。

厳密なメッセージ順序付けや重複排除を維持する必要があるワークロードの場合、Amazon SNS FIFO トピックと、配信エンドポイントとしてサブスクライブされている Amazon SQS FIFO キューを組み合わせると、オペレーションやイベントの順序が重要であるときや重複が許容されないときにアプリケーション間のメッセージングを強化できます。

ベストエフォート型の順序付けと少なくとも 1 回の配信を許容するワークロードでは、Amazon SQS 標準キューを Amazon SNS FIFO トピックにサブスクライブすると、FIFO を利用しないワークロード間でキューを共有できるだけでなく、コストを削減できます。

> **Note**
>
> Amazon SNS FIFO トピックから AWS Lambda 関数へメッセージをファンアウトするには、余分なステップが必要となります。まず、Amazon SQS FIFO キューまたは標準キューをトピックにサブスクライブします。次に、関数をトリガーするようにキューを設定します。詳細については、『AWS コンピューティングブログ』の「イベントソースとしての SQS FIFO」を参照してください。

SNS FIFO トピックでは、E メールアドレス、モバイルアプリケーション、テキストメッセージング (SMS) の電話番号、HTTP (S) エンドポイントなど、顧客マネージドエンドポイントにメッセージを

配信することはできません。これらのエンドポイントタイプは、厳密なメッセージの順序を維持することは保証されません。顧客マネージドエンドポイントを SNS FIFO トピックにサブスクライブしようとすると、エラーが発生します。

SNS FIFO トピックは、標準トピックと同じメッセージフィルター処理機能をサポートします。詳細については、「FIFO トピックのメッセージフィルター処理」および『AWS コンピューティングブログ』の「Amazon SNS メッセージフィルター処理による Pub/Sub メッセージングの簡素化」の投稿を参照してください。

FIFO トピックのメッセージフィルター処理

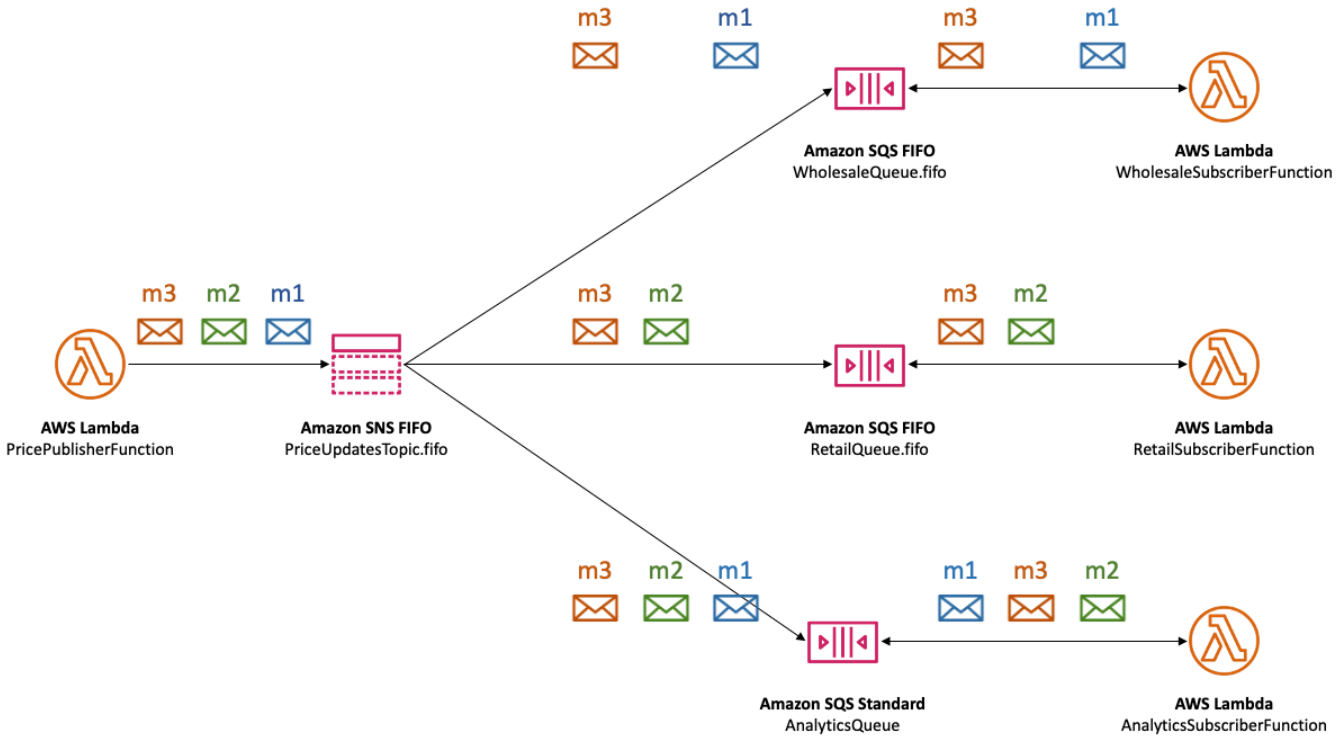
Amazon SNS FIFO トピックでは、メッセージフィルター処理がサポートされています。メッセージのフィルター処理により、受信者システムからのメッセージのルーティングロジックと、受信者システムからのメッセージのフィルター処理ロジックをオフロードすることにより、アーキテクチャが簡素化されます。

Amazon SQS FIFO キューまたは標準キューを SNS FIFO トピックにサブスクライブする場合、メッセージフィルタリングを使用して、サブスクライバーが、すべてのメッセージではなく、メッセージのサブセットを受信するように指定できます。各サブスクライバーは、独自のフィルターポリシーをサブスクリプション属性として設定できます。フィルターポリシーの範囲に基づいて、フィルターポリシーが受信メッセージ属性またはメッセージ本文と照合されます。フィルターポリシーが一致する場合、トピックはメッセージのコピーをサブスクライバーに配信します。一致するものがない場合、トピックはメッセージのコピーを配信しません。

自動車部品価格管理のユースケース例では、以下の Amazon SNS フィルターポリシーが設定されていて、フィルターポリシーの範囲が `MessageBody` であると仮定します。

- 卸売キューについては、フィルターポリシー `{"business":["wholesale"]}` は、`business` という名前のキーと値のセットの `wholesale` を持つすべてのメッセージと一致します。次の図では、メッセージ m1 内のキーの 1 つは `wholesale` の値を持つ `business` です。メッセージ m3 内の 1 つのキーは `["wholesale,retail"]` の値を持つ `business` です。したがって、m1 および m3 の両方がフィルターポリシーの基準に一致し、両方のメッセージが卸売キューに配信されます。
- 小売キューについては、フィルターポリシー `{"business":["retail"]}` は、`business` という名前のキーと値のセットの `retail` を持つすべてのメッセージと一致します。図では、メッセージ m2 内のキーの 1 つは `retail` の値を持つ `business` です。メッセージ m3 内の 1 つのキーは、`["wholesale,retail"]` の値を持つ `business` です。したがって、m2 および m3 の両方がフィルターポリシーの基準に一致し、両方のメッセージが小売キューに配信されます。

- 分析キューでは、Amazon Athena ですべてのレコードを受信したいため、フィルタポリシーは適用しません。



SNS FIFO トピックは、属性文字列値、属性数値、属性キーなど、さまざまなマッチング演算子をサポートします。詳細については、「Amazon SNS メッセージフィルター処理」を参照してください。

SNS FIFO トピックは、サブスクライブされたエンドポイントに重複メッセージを配信しません。詳細については、「FIFO トピックのメッセージ重複除外」を参照してください。

FIFO トピックのメッセージ重複除外

Amazon SNS FIFO トピックおよび Amazon SQS FIFO キューでは、メッセージの重複排除がサポートされます。これは、以下の条件が満たされている限り、メッセージの配信と処理を 1 回だけ行います。

- サブスクライブされた SQS FIFO キューがあり、このキューのアクセス許可により、このキューへのメッセージ配信を Amazon SNS サービスプリンシパルに許可します。

- Amazon SQS FIFO キューコンシューマーはメッセージを処理し、可視性タイムアウトの期限が切れる前に、キューからメッセージを削除します。
- Amazon SNS サブスクリプションとトピックには、メッセージのフィルター処理がありません。メッセージフィルタリングを設定すると、サブスクリプションフィルターポリシーに基づいてメッセージをフィルタリングできるため、Amazon SNS FIFO トピックは at-most-once 配信をサポートします。
- メッセージ配信の確認応答を妨げるネットワークの中断はありません。

> **Note**
>
> メッセージの重複排除は、個別のメッセージグループではなく Amazon SNS FIFO トピック全体に適用されます。

メッセージを Amazon SNS FIFO トピックに発行する場合、メッセージには重複排除 ID を含める必要があります。この ID は、Amazon SNS FIFO トピックから、サブスクライブされた Amazon SQS FIFO キューに配信するメッセージに含まれます。

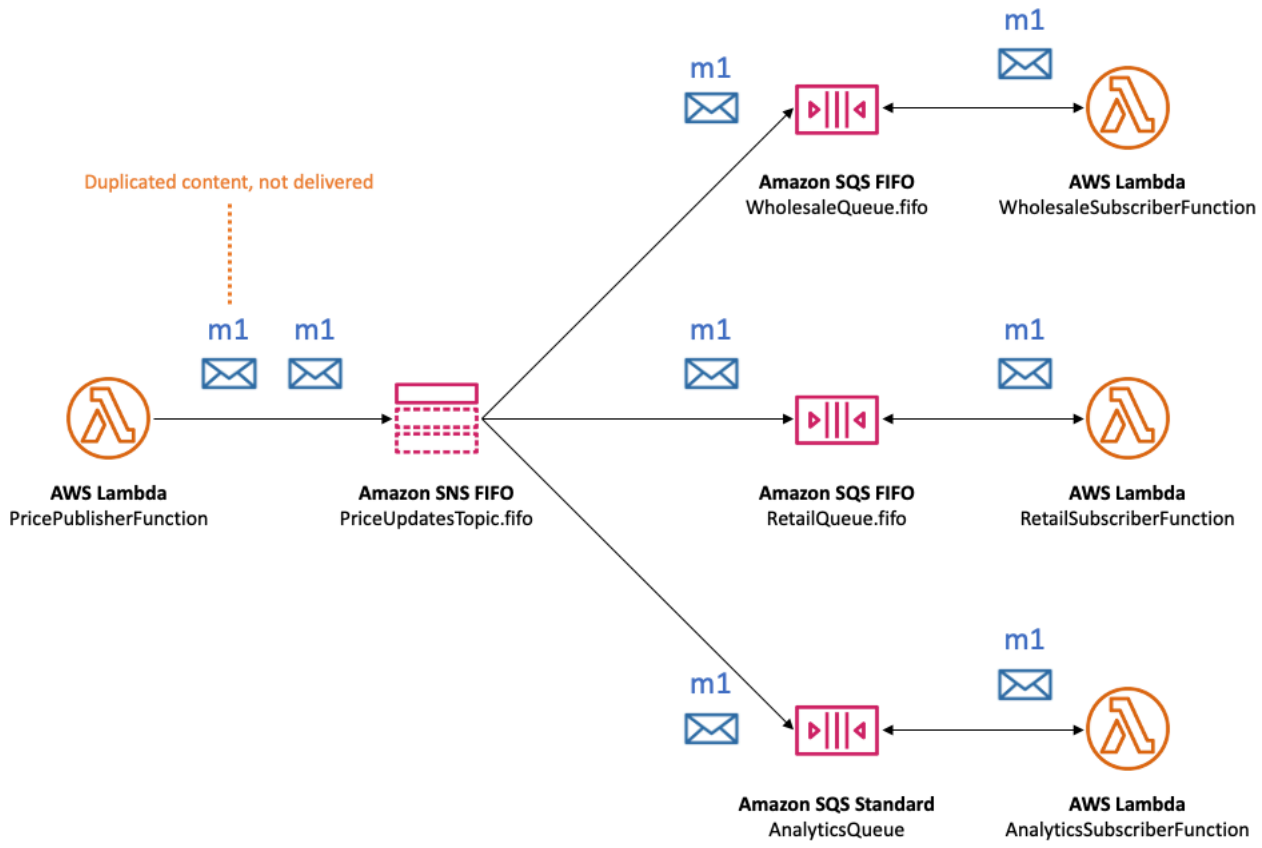
特定の重複排除 ID を持つメッセージが Amazon SNS FIFO トピックに正常に発行された場合、5 分間の重複排除インターバルの間、同じ重複排除 ID で発行されたメッセージは受け付けられますが、配信されません。Amazon SNS FIFO トピックは、サブスクライブされたエンドポイントにメッセージが配信された後も、メッセージの重複排除 ID を追跡し続けます。

メッセージの発行ごとにメッセージ本文が一意であることが保証されている場合は、Amazon SNS FIFO トピックとサブスクライブされた Amazon SQS FIFO キューに対してコンテンツベースの重複排除を有効にできます。Amazon SNS は、メッセージ本文を使用して、各メッセージの重複除外 ID として使用する一意のハッシュ値を生成するため、各メッセージを送信するときに重複除外 ID を設定する必要はありません。

> **Note**
>
> メッセージ属性はハッシュ計算に含まれません。

Amazon SNS FIFO トピックでコンテンツベースの重複排除が有効になっており、メッセージが重複排除 ID で公開されている場合、発行された重複排除 ID は生成されたコンテンツベースの重複排除 ID を上書きします。

自動車部品価格管理のユースケース例では、価格更新ごとに汎用的に一意の重複除外 ID を設定する必要があります。これは、メッセージ属性が卸売と小売で異なる場合でも、メッセージ本文が同一になる可能性があるためです。ただし、企業が、メッセージ本文に製品 ID および製品価格と共にビジネスタイプ (卸売または小売) を追加した場合、Amazon SNS FIFO トピックおよびサブスクライブされた Amazon SQS FIFO キューでコンテンツベースの重複を有効にできます。



メッセージの順序付けと重複排除に加えて、Amazon SNS FIFO トピックは、 AWS KMS キーを使用したメッセージサーバー側の暗号化 (SSE) と、 を使用した VPC エンドポイント経由のメッセージプライバシーをサポートします AWS PrivateLink。詳細については、「FIFO トピックのメッセージセキュリティ」を参照してください。

FIFO トピックのメッセージセキュリティ

AWS Key Management Service(AWS KMS) カスタマーマスターキー (CMK) を使用して、FIFO トピックおよびキューに送信されるメッセージを Amazon SNS および Amazon SQS で暗号化するように選択できます。暗号化された FIFO トピックとキューを作成することも、既存の FIFO トピックとキューを暗号化することもできます。Amazon SNS と Amazon SQS は、メッセージの本文のみを

暗号化します。メッセージ属性、リソースメタデータ、またはリソースメトリックスは暗号化しません。

> **Note**
>
> 既存の FIFO トピックまたはキューに暗号化を追加しても、バックログされたメッセージは暗号化されません。トピックまたはキューから暗号化を削除すると、バックログされたメッセージは暗号化されたままになります。

SNS FIFO トピックは、サブスクライブされたエンドポイントにメッセージを配信する直前にメッセージを復号化します。SQS FIFO キューは、メッセージをコンシューマーアプリケーションに返す直前にメッセージを復号化します。詳細については、「データ暗号化」とAWS コンピューティングブログの「AWS KMS で Amazon SNS に発行したメッセージを暗号化する」の投稿を参照してください。

さらに、SNS FIFO トピックと SQS FIFO キューでは、AWS PrivateLink によるインターフェイス VPC エンドポイントを用いたメッセージプライバシーをサポートしています。インターフェイスエンドポイントを使用すると、公開インターネットを経由することなく、Amazon Virtual Private Cloud (Amazon VPC) サブネットから FIFO トピックおよびキューにメッセージを送信できます。このモデルでは、メッセージは AWS インフラストラクチャとネットワークの中にとどめ、これにより、アプリケーションの全体的なセキュリティが強化されます。AWS PrivateLink を使用すると、インターネットゲートウェイ、ネットワークアドレス変換 (NAT)、またはバーチャルプライベートネットワーク (VPN) を設定する必要はありません。詳細については、「インターネットトラフィックのプライバシー」とAWS コンピューティングブログの「AWS PrivateLink で Amazon SNS に発行したメッセージの確保」の投稿を参照してください。

SNS FIFO トピックは、デッドレターキューとアベイラビリティーゾーン間のメッセージストレージもサポートします。詳細については、「FIFO トピックのメッセージ耐久性」を参照してください。

FIFO トピックのメッセージ耐久性

Amazon SNS FIFO トピックと Amazon SQS キューは耐久性があります。どちらのリソースタイプも、複数のアベイラビリティーゾーンにわたって冗長的にメッセージを保存し、例外的なケースを処理するためのデッドレターキューを提供します。

Amazon SNS では、クライアント側またはサーバー側のエラーにより、Amazon SNS トピックがサブスクライブされた Amazon SQS キューにアクセスできない場合、メッセージの配信は失敗します。

- Amazon SNS FIFO トピックに古いサブスクリプションのメタデータがあると、クライアント側のエラーが発生します。クライアント側のエラーの一般的な 2 つの原因は、Amazon SQS キュー所有者が次のいずれかを行う場合です。
  - キューを削除します。
  - Amazon SNS サービスプリンシパルがメッセージを配信できないようにキューポリシーを変更します。

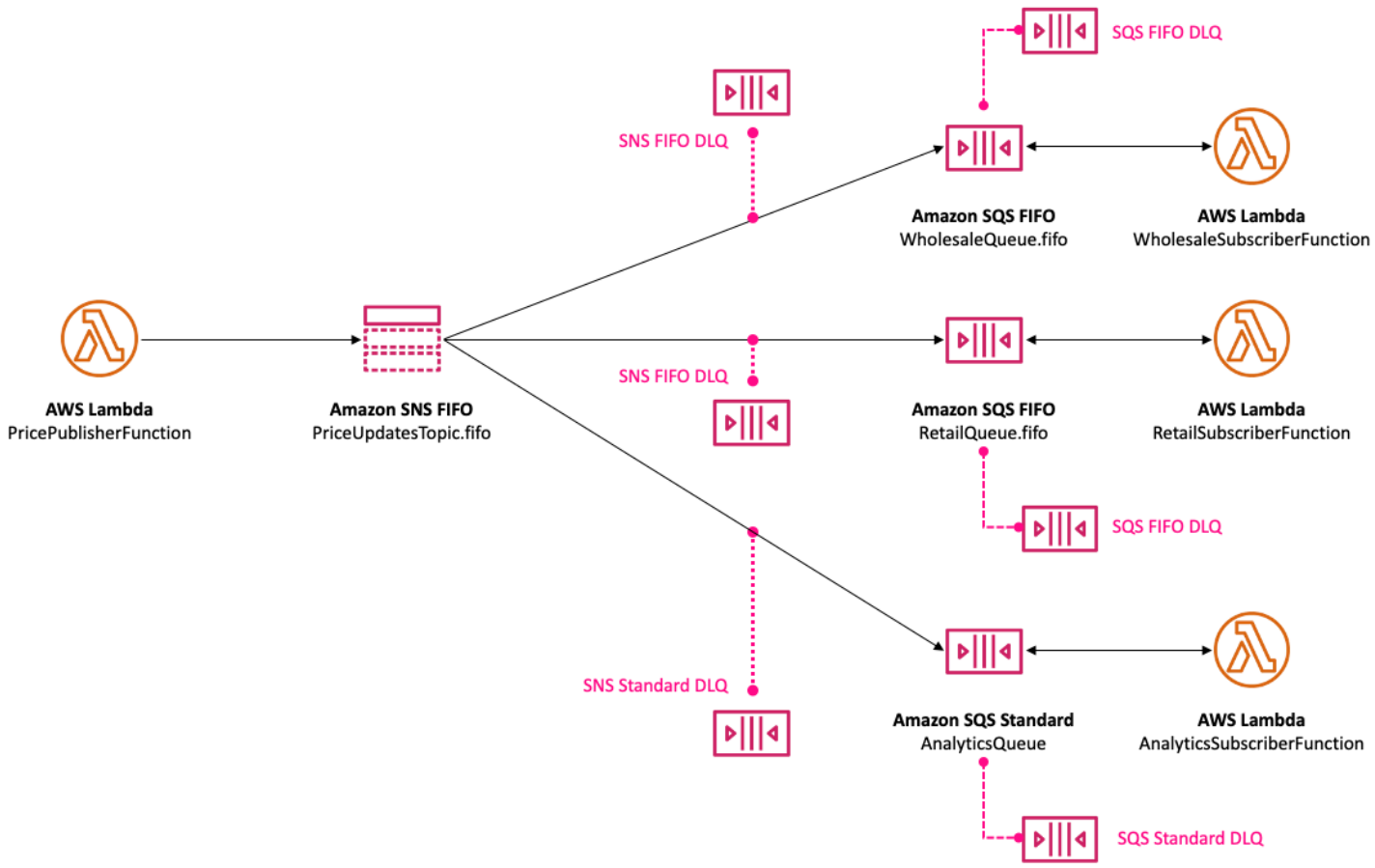
  Amazon SNS は、クライアント側のエラーにより失敗したメッセージの配信を再試行しません。
- サーバー側のエラーは、次の状況で発生する可能性があります。
  - Amazon SQS サービスは利用できません。
  - Amazon SQS は、Amazon SNS サービスからの有効なリクエストを処理できません。

  サーバー側のエラーが発生すると、Amazon SNS FIFO トピックは、失敗した配信を 23 日間、最大 100,015 回再試行します。詳細については、「Amazon SNS メッセージ配信の再試行」を参照してください。

どのようなタイプのエラーでも、Amazon SNS は Amazon SQS デッドレターキューへのメッセージをサイドラインして、データが失われないようにします。

Amazon SQS では、コンシューマーアプリケーションがメッセージの受信、処理、およびキューからの削除に失敗すると、メッセージ処理は失敗します。最大数の受信リクエストが失敗した場合、Amazon SQS はメッセージをデッドレターキューにサイドラインし、データが失われないようにします。

自動車部品価格管理のユースケース例の場合、会社は Amazon SQS デッドレターキュー (DLQ) を各 Amazon SNS FIFO トピックサブスクリプションおよびサブスクライブされた各 Amazon SQS キューに割り当てることができます。これにより、価格更新の損失から会社を保護します。



Amazon SNS サブスクリプションに関連するデッドレターキューは、サブスクライブキューと同じタイプの Amazon SQS キューである必要があります。例えば、Amazon SQS FIFO キューの Amazon SNS FIFO サブスクリプションでは、デッドレターキューとして Amazon SQS FIFO キューが必要です。同様に、Amazon SQS 標準キューの Amazon SNS FIFO サブスクリプションには、デッドレターキューとして Amazon SQS 標準キューが必要です。詳細については、Amazon SNS デッドレターキュー (DLQ) と『AWS コンピューティングブログ』の「Amazon SNS、Amazon SQS 用の DLQ を使用した耐久性のあるサーバーレスアプリケーションの設計、AWS Lambda」の投稿を参照してください。

ダウンストリームの障害からの回復に役立つ耐久性が拡張され、トピック所有者は FIFO トピックを使用してメッセージを最大 365 日間アーカイブすることもできます。トピックサブスクライバーは、これらのメッセージをサブスクライブされたエンドポイントにリプレイして、ダウンストリームアプリケーションの障害によるメッセージを回復したり、既存のアプリケーションの状態を複製したりできます。詳細については、「FIFO トピックのメッセージのアーカイブとリプレイ」を参照してください。

FIFO トピックのメッセージのアーカイブとリプレイ

トピック

メッセージのアーカイブとリプレイとは

Amazon SNS メッセージのアーカイブとリプレイは、トピック所有者がトピック内でメッセージを保存 (またはアーカイブ) できる、ノーコードのインプレースメッセージアーカイブです。トピックサブスクライバーは、アーカイブされたメッセージをサブスクライブされたエンドポイントに対してリトリーブ (またはリプレイ) できます。これを使用して次のことができます。

- ダウンストリームのアプリケーションで障害が発生して失われた可能性があるメッセージを復元する。
- 新しいエンドポイントをサブスクライブし、レプリケート元となるタイムスタンプを選択して、既存のアプリケーションの状態を新しいアプリケーションにレプリケートする。

AWS API、SDK、AWS CloudFormation、および AWS Management Console では、メッセージの Archive and Replay を使用できます。

> **Note**
>
> Amazon SNS メッセージのアーカイブとリプレイは、Application-to-Application (A2A) FIFO トピックでのみ使用できます。

メッセージのアーカイブとリプレイは、次の 2 つの主要コンポーネントで構成されています。

1. メッセージのアーカイブ - トピックの所有者は、トピックのアーカイブとリプレイ機能を有効にし、メッセージの保存期間 (最大 365 日) を設定します。トピック所有者は、Amazon CloudWatch メトリクスを使用してアーカイブされたメッセージをモニタリングすることもできます。詳細については、「FIFO トピック所有者のメッセージアーカイブ」を参照してください。

2. メッセージリプレイ - トピックのサブスクライバーは、トピックからサブスクライブしたエンドポイントへの一連のメッセージのリプレイを開始します。詳細については、「FIFO トピックサブスクライバーのメッセージリプレイ」を参照してください。

FIFO トピック所有者のメッセージアーカイブ

メッセージのアーカイブでは、トピックに発行されたすべてのメッセージの 1 つのコピーをアーカイブできます。トピックのメッセージアーカイブポリシーを有効にすることで、発行されたメッセージをトピック内に保存できます。これにより、そのトピックにリンクされているすべてのサブスクリプションのメッセージアーカイブが有効になります。メッセージのアーカイブ期間は、最短で 1 日、最長で 365 日です。

アーカイブポリシーを設定すると、追加料金がかかります。料金については、「Amazon SNS 料金」を参照してください。

トピック

- AWS Management Console を使用するメッセージアーカイブポリシーを作成する
- API を使用するメッセージアーカイブポリシーを作成する
- SDK を使用するメッセージアーカイブポリシーを作成する
- AWS CloudFormation を使用するメッセージアーカイブポリシーを作成する
- 暗号化されたアーカイブにアクセス権を付与する
- Amazon CloudWatch を使用してメッセージアーカイブメトリクスをモニタリングする

AWS Management Console を使用するメッセージアーカイブポリシーを作成する

AWS Management Console を使用する新しいメッセージアーカイブポリシーを作成するには、このオプションを使用します。

1. Amazon SNS コンソールにサインインします。
2. トピックを選択するか、新しいトピックを作成できます。トピックの作成の詳細については、「Amazon SNS トピックを作成する」を参照してください。

> **Note**
>
> Amazon SNS メッセージのアーカイブとリプレイは、Application-to-Application (A2A) FIFO トピックでのみ使用できます。

3. [トピックの編集] ページで [アーカイブポリシー] セクションを展開します。
4. アーカイブポリシー機能を有効にし、トピック内でメッセージを保存する日数を入力します。
5. [Save changes] (変更の保存) をクリックします。

メッセージアーカイブトピックポリシーを表示、編集、非アクティブ化するには

- トピックの詳細ページでは、設定されている日数などのアーカイブポリシーのステータスがリテンションポリシーに表示されます。[アーカイブポリシー] タブを選択すると、以下のメッセージアーカイブの情報が表示されます。
  - ステータス — アーカイブポリシーが適用されると、アーカイブとリプレイのステータスは [有効] と表示されます。アーカイブポリシーが空の JSON オブジェクトに設定されている場合は、アーカイブとリプレイのステータスは [無効] と表示されます。
  - メッセージ保持期間 — 指定されたメッセージ保持日数。
  - アーカイブ開始日 — サブスクライバーがメッセージを再生できる日付。
  - JSON のプレビュー — アーカイブポリシーの JSON プレビュー。
- (オプション) アーカイブポリシーを編集するには、トピックの [概要] ページに移動して [編集] を選択します。
- (オプション) アーカイブポリシーを無効化するには、トピックの [概要] ページに移動して [編集] を選択します。アーカイブポリシーを無効化し、[変更の保存] を選択します。
- (オプション) アーカイブポリシーを含むトピックを削除するには、前に説明したように、アーカイブポリシーを無効化する必要があります。

> **Important**
>
> メッセージが誤って削除されるのを防ぐため、メッセージアーカイブポリシーが有効なトピックは削除できません。トピックを削除する前に、トピックのメッセージアーカイブポリシーを無効化する必要があります。メッセージアーカイブポリシーを無効にすると、Amazon SNS ではアーカイブされたすべてのメッセージが削除されます。トピックを削除すると、サブスクリプションは削除され、転送中のメッセージが配信されない場合があります。

API を使用するメッセージアーカイブポリシーを作成する

API を使用するメッセージアーカイブポリシーを作成するには、属性 ArchivePolicy をトピックに追加する必要があります。ArchivePolicy は、API アクション CreateTopic と SetTopicAttributes を使用して設定できます。 ArchivePolicy には Amazon SNS がメッセージを保持する日数を表す単一の値 MessageRetentionPeriod が含まれています。トピックのメッセージアーカイブを有効にするには、MessageRetentionPeriod を 0 より大きい整数値に設定します。たとえば、メッセージをアーカイブに 30 日間保持するには、ArchivePolicy を次のように設定します。

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

トピックのメッセージアーカイブを無効にしてアーカイブを消去するには、次のように ArchivePolicy の設定を解除します。

```
{}
```

SDK を使用するメッセージアーカイブポリシーを作成する

AWS SDK を使用するには、認証情報を使用して設定する必要があります。詳細については、『AWS SDK とツールのリファレンスガイド』の「config ファイルおよび credentials ファイル」を参照してください。

次のコード例は、Amazon SNS トピック用にそのトピックに発行されたすべてのメッセージを 30 日間保持するように ArchivePolicy を設定する方法を示しています。

```
// 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);
```

AWS CloudFormation を使用するメッセージアーカイブポリシーを作成する

AWS CloudFormation を使用するアーカイブポリシーを作成するには、『AWS CloudFormation ユーザーガイド』の 「AWS::SNS::Topic」を参照してください。

暗号化されたアーカイブにアクセス権を付与する

暗号化されたトピックからのメッセージのリプレイをサブスクライバーが開始する前に、次の手順を完了する必要があります。過去のメッセージがリプレイされるため、Amazon SNS では、アーカイブ内のメッセージの暗号化に使用された KMS キーへの Decrypt アクセス権がプロビジョニングされている必要があります。

1. KMS キーを使用してメッセージを暗号化し、トピック内に保存する場合、キーポリシーを使用してこれらのメッセージを復号化する機能を Amazon SNS に付与する必要があります。詳細については、「Amazon SNS に復号のアクセス権限を付与する」を参照してください。
2. Amazon SNS に対して AWS KMS を有効にします。詳細については、「AWS KMS 許可を設定する」を参照してください。

> **Important**
>
> 新しいセクションを KMS キーポリシーに追加するときは、ポリシー内の既存のセクションを変更しないでください。トピックの暗号化が有効で、KMS キーが無効か、または削除されている場合、または KMS キーポリシーが Amazon SNS に対して正しく設定されていない場合は、Amazon SNS はサブスクライバーに対してメッセージをリプレイできません。

Amazon SNS に復号のアクセス権限を付与する

Amazon SNS がトピックのアーカイブ内の暗号化されたメッセージにアクセスし、サブスクライブされたエンドポイントにリプレイするには、Amazon SNS サービスプリンシパルでこれらのメッセージを復号する必要があります。

トピック内からの履歴メッセージのリプレイ中に、保存されたメッセージを Amazon SNS サービスプリンシパルが復号することを許可するために必要なポリシーの例を次に示します。

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

Amazon CloudWatch を使用してメッセージアーカイブメトリクスをモニタリングする

Amazon CloudWatch では、以下のメトリックスを使用してアーカイブされたメッセージをモニタリングできます。ワークロードの異常を通知し、影響を回避するのに役立てるために、これらのメトリクスに Amazon CloudWatch アラームを設定できます。詳細については、「Amazon SNS でのログ記録とモニタリング」を参照してください。

メトリクス	説明
ApproximateNumberOfMessagesArchived	トピックアーカイブにアーカイブされたメッセージの総数を 60 分の解像度でトピック所有者に提供します。
ApproximateNumberOfBytesArchived	トピックアーカイブのすべてのメッセージで、アーカイブされたバイトの総数を 60 分の解像度でトピック所有者に提供します。
NumberOfMessagesArchiveProcessing	1 分の解像度のインターバル中に、トピックアーカイブに保存されたメッセージの数をトピックの所有者に通知します。
NumberOfBytesArchiveProcessing	1 分の解像度のインターバル中に、トピックアーカイブに保存されたバイトの総数をトピックの所有者に通知します。

GetTopicAttributes API には、購読者がリプレイを開始できる最も古いタイムスタンプを表す BeginningArchiveTime プロパティが含まれています。以下に、この API アクションのレスポンスの例を示します。

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

FIFO トピックサブスクライバーのメッセージリプレイ

Amazon SNS リプレイでは、トピックサブスクライバーがトピックデータストアからアーカイブされたメッセージを取得し、サブスクライブされたエンドポイントに再配信 (または再生) できます。メッセージは、サブスクリプションが作成されるとすぐにリプレイできます。リプレイされたメッセージには、元のコピーと同じ内容、MessageId、および Timestamp があり、リプレイされたメッセージであることを特定するのに役立つ Replayed 属性も含まれています。特定のメッセージのみをリプレイするために、サブスクリプションにフィルターポリシーを追加できます。メッセージのフィルタリングの詳細については、「リプレイされたメッセージをフィルタリングする」を参照してください。

トピック

- AWS Management Console を使用するメッセージリプレイポリシーを作成する
- API を使用してサブスクリプションにリプレイポリシーを追加する
- SDK を使用してサブスクリプションにリプレイポリシーを追加する
- リプレイされたメッセージをフィルタリングする
- Amazon CloudWatch を使用してメッセージリプレイメトリクスをモニタリングする

AWS Management Console を使用するメッセージリプレイポリシーを作成する

AWS Management Console を使用する新しいリプレイポリシーを作成するには、このオプションを使用します。

1. Amazon SNS コンソールにサインインします。

2. トピックのサブスクリプションを選択するか、新しいトピックを作成できます。サブスクリプションの作成の詳細については、「Amazon SNS トピックへサブスクライブする」を参照してください。
3. メッセージのリプレイを開始するには、[再生] ドロップダウンに移動し、[再生を開始] を選択します。
4. [リプレイ期間] モーダルから、次の選択を行います。
   a. リプレイ開始日時を選択 — アーカイブされたメッセージのリプレイを開始する日付 (YYYY/MM/DD 形式) と時刻 (24 時間の hh:mm:ss 形式) を選択します。開始時刻は、おおよそのアーカイブ時間の開始時刻よりも遅くする必要があります。
   b. (オプション) リプレイ終了日時を選択 — アーカイブされたメッセージのリプレイを停止する日付 (YYYY/MM/DD 形式) と時刻 (24 時間の hh:mm:ss 形式) を選択します。
   c. [Start replay] (再生を開始) を選択します。
5. (オプション) メッセージのリプレイを停止するには、[サブスクリプション] ページに移動し、[再生] ドロップダウンから [リプレイ停止] を選択します。
6. (オプション) CloudWatch を使用してこのワークフロー内からメッセージリプレイメトリクスをモニタリングするには、「Amazon CloudWatch を使用してメッセージリプレイメトリクスをモニタリングする」を参照してください。

メッセージリプレイポリシーを表示および編集するには

[サブスクリプションの詳細] ページでは、次のアクションを実行できます。

- メッセージリプレイステータスについては、[再生ステータス] フィールドに表示される以下の値を確認します。
  - 完了済み — リプレイによりすべてのメッセージが正常に再配信され、現在、新しく発行されたメッセージが配信されています。
  - 進行中 — 現在、選択したメッセージがリプレイ中です。
  - 失敗 — リプレイを完了できませんでした。
  - 保留中 — リプレイ開始時のデフォルト状態です。
- (オプション) メッセージのリプレイを辺クラウドするには、[サブスクリプションの詳細] ページに移動し、[再生] ドロップダウンから [再生を開始] を選択します。リプレイを開始すると、既存のリプレイが置き換えられます。

API を使用してサブスクリプションにリプレイポリシーを追加する

アーカイブされたメッセージをリプレイするには、ReplayPolicy 属性を使用します。ReplayPolicy は、Subscribe および SetSubscriptionAttributes API アクションで使用できます。このポリシーには以下の値があります。

- StartingPoint (必須) — メッセージのリプレイを開始する場所を通知します。
- EndingPoint (オプション) — メッセージのリプレイを停止するタイミングを通知します。EndingPoint を省略すると、現在の時刻に追いつくまでリプレイが続行されます。
- PointType (必須) — 開始ポイントと終了ポイントの種類を設定します。現在、PointType でサポートされている値は Timestamp です。

たとえば、ダウンストリームの障害から回復し、2023 年 10 月 1 日に 2 時間にわたってすべてのメッセージを再送信するには、SetSubscriptionAttributes API アクションを使用して ReplayPolicy を次のように設定します。

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

2023 年 10 月 1 日時点でトピックに送信されたすべてのメッセージをリプレイし、そのトピックに新しく発行されたすべてのメッセージを引き続き受信するには、SetSubscriptionAttributes API アクションを使用してサブスクリプションで ReplayPolicy を次のように設定します。

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

メッセージがリプレイされたことを確認するため、リプレイされた各メッセージに boolean 属性 Replayed が追加されます。

SDK を使用してサブスクリプションにリプレイポリシーを追加する

AWS SDK を使用するには、認証情報を使用して設定する必要があります。詳細については、『AWS SDK とツールのリファレンスガイド』の「config ファイルおよび credentials ファイル」を参照してください。

次のコード例は、Amazon SNS FIFO トピックのアーカイブからのメッセージについて、2023 年 10 月 1 日に 2 時間の時間ウィンドウで再配信するようにサブスクリプションで ReplayPolicy を設定する方法を示しています。

```
// 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);
```

リプレイされたメッセージをフィルタリングする

Amazon SNS メッセージフィルタリングにより、Amazon SNS がサブスクライバーエンドポイントにリプレイするリプレイメッセージを制御できます。メッセージフィルタリングとメッセージアーカイブの両方が有効になっている場合、Amazon SNS は最初にトピックのデータストアからメッセージを取得し、次にサブスクリプションの FilterPolicy に対してメッセージを適用します。メッセージは、一致した場合は、サブスクライブされたエンドポイントに配信され、それ以外の場合は、除外されます。詳細については、「Amazon SNS サブスクリプションフィルターポリシー」を参照してください。

Amazon CloudWatch を使用してメッセージリプレイメトリクスをモニタリングする

Amazon CloudWatch では、以下のメトリックスを使用してリプレイされたメッセージをモニタリングできます。ワークロードの異常を通知し、影響を回避するのに役立てるために、これらのメトリクスに Amazon CloudWatch アラームを設定できます。詳細については、「Amazon SNS での口グ記録とモニタリング」を参照してください。

メトリクス	説明
NumberOfReplayedNotificationsDelivered	1 分の解像度で、トピックアーカイブからリプレイされたメッセージの総数をサブスクライバーに提供します。
NumberOfReplayedNotificationsFailed	1 分の解像度で、リプレイされ、トピックアーカイブからの配信に失敗したメッセージの総数をサブスクライバーに提供します。

FIFO トピックのコード例

以下のコード例に従って、自動車部品価格管理のユースケース例を Amazon SNS FIFO トピックと Amazon SQS FIFO キューまたは標準キューを使用して統合できます。

トピック

- AWS SDK を使用する
- AWS CloudFormation を使用する

AWS SDK を使用する

AWS SDK を使用して、`FifoTopic` 属性を **`true`** に設定することにより、Amazon SNS FIFO トピックを作成します。`FifoQueue` 属性を **`true`** に設定することにより、Amazon SQS FIFO キューを作成します また、**`.fifo`** 接尾辞を各 FIFO リソースの名前に追加する必要があります。FIFO トピックまたはキューを作成した後は、それを標準トピックまたはキューに変換することはできません。

以下のコード例では、これらの FIFO および標準キューリソースを作成します。

- 価格更新を配信する Amazon SNS FIFO トピック
- これらの更新を卸売アプリケーションと小売アプリケーションに提供する Amazon SQS FIFO キュー
- ビジネスインテリジェンス (BI) のクエリが可能なレコードを格納する分析アプリケーション用の Amazon SQS 標準キュー
- 3 つのキューをトピックに接続する Amazon SNS FIFO サブスクリプション

この例では、サブスクリプションでフィルターポリシーを設定します。トピックにメッセージを発行することで、サンプルをテストする場合は、メッセージは必ず business 属性で発行してください。retail または wholesale のいずれかの属性値を指定します。それ以外を指定すると、メッセージは除外され、サブスクライブされたキューに配信されません。詳細については、「FIFO トピックのメッセージフィルター処理」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

この例では

- 1 つの Amazon SNS FIFO トピック、2 つの Amazon SQS FIFO キュー、および 1 つの標準キューを作成します。
- キューをトピックにサブスクライブし、メッセージをトピックに発行します。

テストでは、各キューへのメッセージの受信を検証します。完全な例では、アクセスポリシーの追加と、最後にリソースの削除も示しています。

```
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);
       }
   }
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - CreateTopic

- Publish
- Subscribe

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

FIFO トピックを作成し、そのトピックに Amazon SQS FIFO キューと標準キューをサブスクライブして、メッセージを Amazon SNS トピックに発行します。

```
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
```

- API の詳細については、『AWS SDK for Python (Boto3) API リファレンス』の以下のトピックを参照してください。
  - CreateTopic
  - Publish
  - Subscribe

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

FIFO トピックを作成し、そのトピックに Amazon SQS FIFO キューをサブスクライブして、Amazon SNS トピックにメッセージを発行します。

```
    " 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.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の以下のトピックを参照してください。
  - CreateTopic
  - Publish
  - Subscribe

FIFO サブスクリプションからメッセージを受信する

3 つのサブスクライブされたアプリケーションで価格の更新を受け取ることができるようになりました。the section called “FIFO トピックのユースケース” に示すように、各コンシューマーアプリケーションのエントリーポイントは Amazon SQS キューであり、対応する AWS Lambda 関数から自動的にポーリングできます。Amazon SQS キューが Lambda 関数のイベントソースである場合、Lambda はメッセージを効率的に使用するために、必要に応じてポーラーのフリートをスケールします。

詳細については、『AWS Lambda デベロッパーガイド』の「Amazon SQS で AWS Lambda 使用する」を参照してください。独自のキューポーラーの書き込み方法については、『Amazon Simple Queue Service デベロッパーガイド』の「Amazon SQS スタンダードおよび FIFO キューのレコメンデーション」および『Amazon Simple Queue Service API リファレンス』の「ReceiveMessage」を参照してください。

AWS CloudFormation を使用する

AWS CloudFormation では、テンプレートファイルを使用して、AWS リソースのコレクションを単一のユニットとして作成および設定できます。このセクションでは、以下を作成するサンプルテンプレートを使用します。

- 価格更新を配信する Amazon SNS FIFO のトピック
- これらの更新を卸売アプリケーションと小売アプリケーションに提供する Amazon SQS FIFO キュー
- ビジネスインテリジェンス (BI) のクエリが可能なレコードを格納する分析アプリケーション用の Amazon SQS 標準キュー
- 3 つのキューをトピックに接続する Amazon SNS FIFO サブスクリプション
- 受信者アプリケーションが必要とする価格の更新のみを受信することを指定するフィルターポリシー

> **Note**
>
> トピックにメッセージを発行することで、サンプルをテストする場合は、メッセージは必ず business 属性で発行してください。retail または wholesale のいずれかの属性値を指定します。それ以外を指定すると、メッセージは除外され、サブスクライブされたキューに配信されません。

```
{
  "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"
          }
        ]
      }
    }
  }
}
```

AWS CloudFormation テンプレートを使用した AWS リソースのデプロイの詳細については、『AWS CloudFormation ユーザーガイド』の「開始方法」を参照してください。

Amazon SNS メッセージの発行

Amazon SNS トピックを作成して、このトピックにエンドポイントをサブスクライブしたら、トピックにメッセージを発行できます。メッセージが公開されると、Amazon SNS はサブスクライブされたエンドポイントにメッセージを配信しようとします。

トピック

- AWS Management Consoleを使用してAmazon SNS トピックにメッセージを発行するには
- AWS SDK を使用してトピックにメッセージを発行するには
- Amazon SNS および Amazon S3 を使用した容量の大きなメッセージを発行する
- Amazon SNS メッセージ属性
- Amazon SNS メッセージのバッチ処理

AWS Management Consoleを使用してAmazon SNS トピックにメッセージを発行するには

1. Amazon SNS コンソールにサインインします。
2. 左のナビゲーションペインで、[トピック] を選択します。
3. [トピック] ページで、トピックを選択し、[メッセージの発行] を選択します。

   コンソールが [トピックにメッセージを発行] ページを開きます。
4. [メッセージ詳細] セクションで、次の操作を行います。

   a. (オプション) メッセージの件名を入力します。

   b. FIFO トピックは、メッセージグループ ID を入力します。同じメッセージグループ内のメッセージは、発行された順序で配信されます。

   c. FIFO トピックは、メッセージ重複排除 ID を入力します。トピックの[コンテンツベースのメッセージ重複除外] 設定を有効にしていれば、この ID は任意です。

   d. (オプション) モバイルプッシュ通知に、有効期限 (TTL) 値を秒単位で入力します。この属性で、Apple Push Notification Service (APNs) や Firebase Cloud Messaging (FCM) などのプッシュ通知サービスによってエンドポイントにメッセージが配信される時間を指定できます。
5. [メッセージ本文] セクションで、以下のいずれかの操作を行います。

a. [すべての配信プロトコルに同一のペイロード] を選択し、メッセージを入力します。

b. [各配信プロトコルのカスタムペイロード] を選択し、JSON オブジェクトを入力し、各プロトコルに送信するメッセージを定義します。

詳細については、「プラットフォーム固有のペイロードによる公開」を参照してください。

6. [メッセージの属性] セクションで、属性を追加します。Amazon SNS は、追加された属性とサブスクリプション属性 `FilterPolicy` をマッチングし、サブスクライブされたエンドポイントが、発行されたメッセージに関心があるかどうかを判断します。

a. タイプでは、String.Array などの属性タイプを選択します。

> **Note**
>
> 属性のタイプが String.Array である場合は、配列を角括弧 (`[]`) で囲みます。配列内で、値の文字列を二重引用符で囲みます。数字またはキーワード `true`、`false`、`null` を引用符で囲む必要はありません。

b. `customer_interests` などの属性の名前を入力します。

c. `["soccer", "rugby", "hockey"]` などの属性の値を入力します。

属性のタイプが、String、String.Array、または Number である場合、フィルターポリシー (存在する場合) の範囲が明示的に `MessageBody` に設定されていなければ、Amazon SNS はサブスクリプションにメッセージを送信する前に、サブスクリプションのフィルターポリシーに照らしてメッセージ属性を評価します。

詳細については、「Amazon SNS メッセージ属性」を参照してください。

7. [メッセージの発行] を選択します。

トピックにメッセージが発行され、コンソールがトピックの [詳細] ページを開きます。

AWS SDK を使用してトピックにメッセージを発行するには

AWS SDK を使用するには、認証情報を使用して設定する必要があります。詳細については、「AWS SDK とツールのリファレンスガイド」の「共有設定ファイルと認証情報ファイル」を参照してください。

次のコード例は使用方法を示しています`Publish`。

.NET

AWS SDK for .NET

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックへのメッセージの発行

```
    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}");
        }
    }
```

グループ、重複、属性のオプションを指定してメッセージをトピックに発行します。

```
    /// <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);
        }
    }
```

ユーザーの選択を発行アクションに適用します。

```
    /// <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;
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「Publish」を参照してください。

C++

SDK for C++

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

属性付きのメッセージを公開する。

```
        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;
        }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「Publish」を参照してください。

CLI

AWS CLI

例 1: トピックにメッセージを発行するには

次の publish の例では、指定した Amazon SNS トピックに指定した通知を公開します。メッセージはテキストファイルから取得されたもので、改行を含めることができます。

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

message.txt の内容：

```
Hello World
Second Line
```

出力:

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

例 2: 電話番号に SMS メッセージを公開するには

次の publish の例では、Hello world! メッセージを電話番号 +1-555-555-0100 に公開します。

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「Publish」を参照してください。

Go

SDK for Go V2

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「発行」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「Publish」を参照してください。

JavaScript

JavaScript (v3) 用の SDK

> **Note**
>
> にはまだまだあります。 GitHub用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

グループ、重複、属性のオプションを指定してメッセージをトピックに発行します。

```
  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();
    }
  }
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for JavaScript API リファレンスの「発行」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Publish」を参照してください。

PHP

SDK for PHP

> **Note**
>
> まだまだあります GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API リファレンスの「発行」を参照してください。

PowerShell

以下のためのツール PowerShell

例 1: この例は、 MessageAttribute インラインで宣言された 1 つのメッセージを含むメッセージの公開を示しています。

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

例 2: この例は、 MessageAttributes 複数を事前に宣言したメッセージを公開する例です。

```
$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
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス」の「Publish」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> にはまだまだあります。 GitHubAWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

サブスクリプションが属性に基づいてフィルター処理できるように、属性を含むメッセージを発行します。

```
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
```

受信者のプロトコルに基づいて異なる形式のメッセージを発行します。

```
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
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「Publish」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> まだまだあります GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for Ruby API リファレンスの「発行」を参照してください。

Rust

SDK for Rust

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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(())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「発行」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの「Publish」(発行) を参照してください。

Amazon SNS および Amazon S3 を使用した容量の大きなメッセージを発行する

容量の大きな Amazon SNS メッセージを発行するには、Java 用 Amazon SNS 拡張クライアントライブラリまたは Python 用 Amazon SNS 拡張クライアントライブラリを使用できます。このライブラリは、現在の最大容量 256 KB を超える、最大 2 GB までのメッセージに便利です。両方のライブラリは、実際のペイロードを Amazon S3 バケットに保存し、保存された Amazon S3 オブジェクトの参照をトピックに発行します。サブスクライブした Amazon SQS キューは、Java 用 Amazon SQS 拡張クライアントライブラリを使用し、Amazon S3 からペイロードを逆参照して取得します。Lambda などの他のエンドポイントは、Payload Offloading Java Common Library for AWS を使用し、ペイロードを逆参照して取得します。

> **Note**
>
> Amazon SNS 拡張クライアントライブラリは、標準トピックと FIFO トピックの両方と互換性があります。

トピック

Java 用の拡張クライアントライブラリ

トピック

前提条件

Java 用 Amazon SNS 拡張クライアントライブラリを使用するための前提条件は以下のとおりです。

- AWS SDK。

このページの例では、 AWS Java SDK を使用しています。SDK をインストールしてセットアップするには、「AWS SDK for Java デベロッパーガイド」の「AWS SDK for Java のセットアップ」を参照してください。

- 適切な認証情報 AWS アカウント を持つ 。

  を作成するには AWS アカウント、AWS ホームページ に移動し、アカウントの作成 AWS を選択します。手順に従います。

  認証情報の詳細については、「 AWS SDK for Java デベロッパーガイド」の「開発用の AWS 認証情報とリージョンのセットアップ」を参照してください。
- Java 8 以上
- Java 用 Amazon SNS 拡張クライアントライブラリ (Maven からも利用可能)

メッセージストレージの設定

Amazon SNS 拡張クライアントライブラリは、メッセージの保存と取得に Payload Offloading Java Common Library AWS を使用します。以下の Amazon S3 メッセージストレージオプションを設定できます。

- カスタムメッセージサイズのしきい値 — このサイズを超えるペイロードと属性を持つメッセージは、自動的に Amazon S3 に保存されます。
- `alwaysThroughS3` フラグ — この値を `true` に設定し、すべてのメッセージペイロードを強制的に Amazon S3 に保存します。例:

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

- カスタム KMS キー — Amazon S3 バケットのサーバー側の暗号化に使用するキー。
- バケット名 — メッセージペイロードを保存するための Amazon S3 バケットの名前。

例: Amazon S3 に保存されたペイロードを使用して Amazon SNS にメッセージを発行する

次のコードサンプルは、以下の操作方法を示しています。

- サンプルのトピックとキューを作成します。

- トピックからメッセージを受信するためにキューをサブスクライブします。
- テストメッセージを発行します。

メッセージペイロードは Amazon S3 に保存され、そのペイロードへのリファレンスが発行されます。メッセージの受信には、Amazon SQS 拡張クライアントが使用されます。

SDK for Java 1.x

> **Note**
>
> には他にもがあります GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

容量の大きなメッセージを発行するには、Amazon SNS Extended Client Library for Java を使用します。送信するメッセージは、実際のメッセージコンテンツを含む Amazon S3 オブジェクトをリファレンスします。

```
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());
        }
}
```

その他のエンドポイントプロトコル

Amazon SNS と Amazon SQS ライブラリの両方で、Payload Offloading Java Common Library for AWS を使用して、Amazon S3 でメッセージペイロードを保存および取得できます。Java が有効な

エンドポイント (Java で実装されている HTTPS エンドポイントなど) は、同じライブラリを使用してメッセージコンテンツを逆リファレンスできます。

のペイロードオフロード Java 共通ライブラリを使用できないエンドポイントは、Amazon S3 に保存されているペイロードを含むメッセージを引き続き発行 AWS できます。以下は、上記のコード例で発行された Amazon S3 リファレンスの例です。

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

Python 用の拡張クライアントライブラリ

トピック

- 前提条件
- メッセージストレージの設定
- 例: Amazon S3 にペイロードが保存されている Amazon SNS にメッセージを発行する

前提条件

Python 用 Amazon SNS 拡張クライアントライブラリを使用するための前提条件は以下のとおりです。

- AWS SDK。

  このページの例では、 AWS Python SDK Boto3 を使用しています。SDK をインストールしてセットアップするには、AWS SDK for Python のドキュメントを参照してください。
- 適切な認証情報 AWS アカウント を持つ 。

  を作成するには AWS アカウント、AWS ホームページ に移動し、アカウントの作成 AWS を選択します。手順に従います。

  認証情報については、「AWS SDK for Python デベロッパーガイド」の「認証情報」を参照してください。
- Python 3.x (または以降) および pip。

- Python 用 Amazon SNS 拡張クライアントライブラリ (PyPI からも利用可能)

メッセージストレージの設定

以下の属性は、Amazon S3 メッセージストレージオプションを設定するための Boto3 Amazon SNS クライアント、トピック 、および PlatformEndpoint オブジェクトで使用できます。 Amazon S3

- `large_payload_support` - サイズの大きいメッセージを保存する Amazon S3 バケットの名前。
- `message_size_threshold` - メッセージを大きなメッセージバケットに保存するためのしきい値。0 未満または 262144 を超える値を指定することはできません。デフォルトは 262144 です。
- `always_through_s3` - `True` の場合、すべてのメッセージは Amazon S3 に保存されます。デフォルトは `False` です。
- `s3` - Amazon S3 にオブジェクトを保存するために使用する Boto3 Amazon S3 の `resource` オブジェクト。Amazon S3 リソース (Amazon S3 カスタム設定や認証情報など) を制御する場合に使用します。初回使用時に設定しない場合、デフォルトは `boto3.resource("s3")` です。

例: Amazon S3 にペイロードが保存されている Amazon SNS にメッセージを発行する

次のコード例は、以下の操作方法を示しています。

- Amazon SNS トピックと Amazon SQS キューを作成します。
- トピックからメッセージを受信するためにキューをサブスクライブします。
- テストメッセージを発行します。
- メッセージペイロードは Amazon S3 に保存され、その参照が発行されます。
- キューから発行したメッセージと、Amazon S3 から取得した元のメッセージを印刷します。

容量の大きなメッセージを発行するには、Java 用 Amazon SNS 拡張クライアントライブラリを使用します。送信するメッセージは、実際のメッセージコンテンツが含まれている Amazon S3 オブジェクトを参照します。

```
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 client
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)
```

出力

```
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 メッセージ属性

Amazon SNS では、メッセージに関する構造化メタデータ項目 (タイムスタンプ、地理空間データ、署名、識別子など) を指定できるメッセージ属性の配信をサポートしています。SQS サブスクリプションの場合、raw メッセージの配信を有効にすると、最大 10 個のメッセージ属性を送信できます。10 個を超えるメッセージ属性を送信するには、Raw メッセージの配信を無効にする必要があります。raw メッセージ配信が有効になっている Amazon SQS サブスクリプション宛てのメッセージ属性が 10 個を超えるメッセージは、クライアント側のエラーとして破棄されます。

メッセージ属性はオプションであり、メッセージ本文とは別個のものですが、同時に送信されます。受信者は、この情報を使用して、メッセージ本文を最初に処理する必要なしでメッセージを処理する方法を決定できます。

AWS Management Console または AWS SDK for Java を使用して、属性を使ってメッセージを送信する詳細については、「AWS Management Consoleを使用してAmazon SNS トピックにメッセージを発行するには」チュートリアルを参照してください。

> **Note**
>
> メッセージ属性は、メッセージ構造が JSON ではなく String である場合にのみ送信されます。

また、モバイルエンドポイント用のプッシュ通知メッセージを構築するためにメッセージ属性を使用することもできます。このシナリオでは、メッセージ属性はプッシュ通知メッセージの構築のみに使用されます。属性は、Amazon SQS エンドポイントにメッセージ属性とともにメッセージを送信する場合とは異なり、エンドポイントには配信されません。

メッセージ属性を使用して、サブスクリプションフィルターポリシーでメッセージをフィルター処理可能にすることもできます。フィルターポリシーは、トピックのサブスクリプションに適用もできます。フィルターポリシーの範囲を `MessageAttributes` (デフォルト) に設定して、フィルターポリシーを適用すると、サブスクリプションは、ポリシーが受け入れる属性を持つメッセージのみを受信します。詳細については、「Amazon SNS メッセージフィルター処理」を参照してください。

> **Note**
>
> メッセージ属性をフィルタリングに使用する場合、値は有効な JSON 文字列でなければなりません。これにより、メッセージ属性フィルタリングが有効になっているサブスクリプションにメッセージが確実に配信されます。

メッセージ属性の項目および検証

各メッセージ属性は、次の項目で構成されています。

- 名前 - メッセージ属性名には、A-Z、a-z、0-9、下線 (_)、ハイフン (-)、ピリオド (.) を使用できます。名前の先頭と末尾をピリオドにすることはできず、ピリオドを連続して使用することはできません。名前では大文字と小文字が区別され、メッセージのすべての属性名間で一意にする必要があります。名前の長さは最大 256 文字です。名前の先頭を「`AWS.`」や「`Amazon.`」(または、大文字と小文字が異なるあらゆる変化形) にすることはできません。これらのプレフィックスは Amazon Web Services で使用するために取り置かれています。
- 型 - サポートされるメッセージ属性のデータ型は、`String`、`String.Array`、`Number`、`Binary`です。データ型のコンテンツには、メッセージ本文と同じ制限があります。データ型では大文字と小文字が区別され、長さは最大 256 バイトです。詳細については、「メッセージ属性のデータ型と検証」セクションを参照してください。

- 値 - ユーザー指定のメッセージ属性値。文字列データ型の場合、値属性のコンテンツにはメッセージ本文と同じ制限があります。詳細については、『Amazon Simple Notification Service API リファレンス』の「公開アクション」を参照してください。

名前、型、値を空または Null にすることはできません。さらに、メッセージ本文を空または Null にすることもできません。メッセージ属性のすべての部分 (名前、型、値を含む) は、メッセージサイズの制限に含められます。制限は 256 KB です。

メッセージ属性のデータ型と検証

メッセージ属性のデータ型は、メッセージ属性が Amazon SNS によって処理される方法を特定します。例えば、型が数値の場合、Amazon SNS はその属性が数値であることを検証します。

Amazon SNS は、記載されている場合を除き、すべてのエンドポイントについて、次の論理データ型をサポートしています。

- 文字列 - 文字列は、UTF-8 バイナリエンコードされた Unicode です。コードの値のリストについては、http://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters を参照してください。

  > **Note**
  >
  > サロゲート値は、メッセージ属性ではサポートされていません。例えば、サロゲート値を使用して絵文字を表すと、次のエラーが発生します: `Invalid attribute value was passed in for message attribute`。

- String.Array - 複数の値を含むことができる文字列として書式設定される配列。この値は、文字列、数値、またはキーワード (`true`、`false`、`null`) とすることができます。数値型またはブール型の String.Array は引用符を必要としません。複数の String.Array 値はカンマで区切られます。

  このデータ型は AWS Lambda のサブスクリプションのサポート外です。このデータ型を Lambda エンドポイントに指定すると、`String` データ型を、Amazon SNS が Lambda に配信する JSON ペイロードに格納します。

- 数値 - 数値は正または負の整数か、浮動小数点数です。数値には、整数、浮動小数点数、倍精度浮動小数点数が通常サポートするほとんどの値を包含できる十分な範囲と精度があります。数値は -10^9 ~ 10^9 までの値とすることができ、小数点以下 5 桁の精度を持ちます。先頭と末尾の 0 は切り捨てられます。

このデータ型は AWS Lambda のサブスクリプションのサポート外です。このデータ型を Lambda エンドポイントに指定すると、`String` データ型を、Amazon SNS が Lambda に配信する JSON ペイロードに格納します。

- Binary - Binary 型の属性には、圧縮データ、暗号化データ、イメージなど、任意のバイナリデータが保存されます。

モバイルプッシュ通知の予約済みメッセージ属性

次の表は、プッシュ通知メッセージを構築するために使用できるモバイルプッシュ通知サービスの予約済みメッセージ属性の一覧です。

プッシュ通知サービス	予約済みメッセージ属性
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`

プッシュ通知サービス	予約済みメッセージ属性
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
	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] メッセージ属性が要件を満たさない場合、Apple は Amazon SNS 通知を拒否します。詳細については、Apple Developer ウェブサイトの「APNs への通知リクエストの送信」を参照してください。

Amazon SNS メッセージのバッチ処理

メッセージのバッチ処理とは何ですか。

個別の Publish API リクエストで標準トピックまたは FIFO トピックにメッセージを発行する代わりに、Amazon SNS PublishBatch API を使用して、単一の API リクエストで最大 10 のメッセージを発行します。メッセージをバッチで送信すると、分散アプリケーションの接続 (A2A メッセージング) または Amazon SNS を使用した人への通知の送信 (A2P メッセージング) に関連するコストを最大 10 分の 1 に削減できます。Amazon SNS では、操作するリージョンに基づいて、1 秒間にトピックに発行できるメッセージの数に関するクォータがあります。API クォータの詳細については、「AWS 全般のリファレンス」の「Amazon SNS エンドポイントとクォータ」のページを参照してください。

> **Note**
>
> 1 回の PublishBatch API 呼び出しで送信するすべてのメッセージの合計サイズは 262,144 バイト (256 KB) を超えることはできません。
> PublishBatch API は、IAM ポリシーの同じ Publish API アクションを使用します。

メッセージのバッチ処理はどのように機能しますか。

Publish API を使用したメッセージの公開は、PublishBatch API を使用したメッセージの公開に似ています。主な違いは、PublishBatch API リクエストでは、各メッセージに一意のバッチ ID (最大 80 文字) を割り当てる必要があります。このようにして、Amazon SNS は、バッチ内のすべてのメッセージに対して個々の API レスポンスを返して、各メッセージが発行されたか、または障害が発生したかを確認できます。FIFO トピックに発行されるメッセージについては、一意のバッチ ID の割り当てに加えて、個々のメッセージごとに MessageDeduplicationID および MessageGroupId を含める必要があります。

例

10 のメッセージのバッチを標準トピックに公開する

```
// 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);
    }
}
```

10 のメッセージのバッチを FIFO トピックに公開する

```
// 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);
    }
}
```

Amazon SNS メッセージフィルター処理

デフォルトでは、Amazon SNS トピックの受信者は、トピックに対して発行されたすべてのメッセージを受信します。メッセージのサブセットのみを受信する場合、受信者はトピックのサブスクリプションにフィルターポリシーを割り当てる必要があります。

フィルターポリシーは、受信者が受信するメッセージを定義するプロパティが含まれている JSON オブジェクトです。Amazon SNS は、サブスクリプションに設定したフィルターポリシーの範囲に従って、メッセージ属性またはメッセージ本文に適用されるポリシーに対応します。メッセージ本文のフィルターポリシーは、メッセージペイロードが正しい形式の JSON オブジェクトであることを前提としています。

サブスクリプションにフィルターポリシーがない場合、サブスクライバーは、そのトピックに発行されたすべてのメッセージを受信します。フィルターポリシーを設定してトピックにメッセージを発行すると、Amazon SNS はメッセージ属性またはメッセージ本文をトピックの各サブスクリプションのフィルターポリシー内のプロパティを比較します。一致するメッセージ属性またはメッセージ本文プロパティがある場合、Amazon SNS は受信者にメッセージを送信します。そうでない場合、Amazon SNS はそのサブスクライバーにメッセージを送信しません。

詳細については、「トピックに発行されたメッセージのフィルター処理」を参照してください。

トピック

- Amazon SNS サブスクリプションフィルターポリシーの範囲
- Amazon SNS サブスクリプションフィルターポリシー
- サブスクリプションフィルターポリシーを適用する
- サブスクリプションフィルターポリシーを削除する

Amazon SNS サブスクリプションフィルターポリシーの範囲

`FilterPolicyScope` サブスクリプション属性では、次のいずれかの値を設定してフィルタリング範囲を選択できます。

- `MessageAttributes` — フィルターポリシーがメッセージ属性に適用されます。これがデフォルトです。
- `MessageBody` — フィルターポリシーがメッセージ本文に適用されます。

> **Note**
>
> 既存のフィルターポリシーに範囲が定義されていない場合、範囲はデフォルトで `MessageAttributes` に設定されます。

Amazon SNS サブスクリプションフィルターポリシー

サブスクリプションフィルターポリシーを使用すると、プロパティ名を指定して、プロパティ名ごとに値のリストを割り当てることができます。詳細については、「Amazon SNS メッセージフィルター処理」を参照してください。

Amazon SNS がサブスクリプションフィルターポリシーに照らしてメッセージ属性またはメッセージ本文のプロパティを評価する際、ポリシーで指定されていないものは無視されます。

> **Important**
>
> AWS IAM や Amazon SNS などの サービスは、結果整合性と呼ばれる分散コンピューティングモデルを使用します。サブスクリプションフィルターポリシーへの追加または変更は、完全に有効になるまでに最大 15 分かかります。

サブスクリプションは、以下の条件に該当するメッセージを受け入れます。

- フィルターポリシーの範囲が `MessageAttributes` に設定されている場合、フィルターポリシーの各プロパティ名はメッセージ属性名と一致します。フィルターポリシー内で一致するプロパティ名ごとに、少なくとも 1 つのプロパティ値がメッセージ属性値と一致します。
- フィルターポリシーの範囲が `MessageBody` に設定されている場合、フィルターポリシーの各プロパティ名はメッセージ本文のプロパティ名と一致します。フィルターポリシー内で一致するプロパティ名ごとに、少なくとも 1 つのプロパティ値がメッセージ本文のプロパティ値と一致します。

Amazon SNS は、現在、以下のフィルター演算子をサポートしています。

- AND ロジック
- OR ロジック
- OR 演算子

- キーの一致
- 数値の完全一致
- 数値の anything-but 一致
- 数値範囲の一致
- 文字列値の完全一致
- 文字列値の anything-but 一致
- プレフィックスと anything-but 演算子を使用した文字列の一致
- 文字列値の equals-ignore case
- 文字列値の IP アドレス一致
- 文字列値のプレフィックスマッチング
- 文字列値のサフィックスマッチング

フィルターポリシーの例

次の例は、顧客のトランザクションを処理する Amazon SNS トピックから配信されるメッセージペイロードを示しています。

最初の例には、トランザクションを記述する属性がある `MessageAttributes` フィールドが含まれます。

- 顧客の興味
- ストア名
- イベント状態
- 購入価格 (USD)

このメッセージには `MessageAttributes` フィールドが含まれるため、サブスクリプションで `FilterPolicyScope` が `MessageAttributes` に設定されている限り、`FilterPolicy` を設定するトピックサブスクリプションは、メッセージを選択的に許可または拒否することができます。メッセージへの属性の適用の詳細については、「Amazon SNS メッセージ属性」を参照してください。

```
{
   "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"
      }
   }
}
```

次の例は、Message フィールドに含まれ、メッセージペイロードまたはメッセージ本文とも呼ばれる同じ属性を示しています。サブスクリプションで FilterPolicyScope が MessageBody に設定されている限り、FilterPolicy を含むトピックサブスクリプションは、メッセージを選択的に許可または拒否することができます。

```
{
"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"
```

```
}
```

以下のフィルターポリシーでは、プロパティの名前と値に基づいてメッセージを許可または拒否します。

メッセージ例を許可するポリシー

以下のサブスクリプションフィルターポリシーのプロパティは、メッセージ例に割り当てられた属性に一致します。MessageAttributes または MessageBody に設定されているかどうかにかかわらず、同じフィルターポリシーが FilterPolicyScope に対して有効であることに注意してください。各サブスクライバーは、トピックから受信するメッセージの構成に従ってフィルタリング範囲を選択します。

このポリシーの 1 つのプロパティが、メッセージに割り当てられた属性と一致しなかった場合、ポリシーはメッセージを拒否します。

```
{
   "store": ["example_corp"],
   "event": [{"anything-but": "order_cancelled"}],
   "customer_interests": [
      "rugby",
      "football",
      "baseball"
   ],
   "price_usd": [{"numeric": [">=", 100]}]
}
```

メッセージ例を拒否するポリシー

以下のサブスクリプションフィルターポリシーでは、そのプロパティと、メッセージ例に割り当てられた属性の間に、複数の不一致があります。例えば、encrypted プロパティ名はメッセージ属性に存在していないため、割り当てられている値にかかわらず、このポリシープロパティによってメッセージは拒否されます。

不一致が発生すると、ポリシーはメッセージを拒否します。

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

```
      "basketball",
      "baseball"
   ]
}
```

フィルターポリシーの制約

Amazon SNS サブスクリプションのフィルターポリシーを作成するときは、ポリシー内のキーのカウント方法を理解することが重要です。覚えておくべき重要な側面は次のとおりです。

1. 親キー – 親キーは、フィルターポリシーの最上位キーです。これらは、値または制約を指定するキーです。
2. 属性名 – 親キーは、フィルターポリシーの属性名と見なされます。これらのキーに指定した値または制約は、メッセージペイロードの対応する属性に適用されます。
3. 有効な値 – 親キーに指定される値は、文字列、文字列の配列、または数値のいずれかである必要があります。値がオブジェクト (JSON オブジェクトなど) の場合、フィルターポリシーでは有効なキーとしてカウントされません。

次のフィルターポリシーの例を考えてみましょう。

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

この例では、次のキーがフィルターポリシーの一部としてカウントされます。

- `state`
- `severity`
- `message`
- `standard_control`
- `region`

- account

キーの検出結果は、文字列、文字列の配列、または数値ではなく、JSON オブジェクトがその値として含まれるためカウントされません。

別の例を紹介します。

```
{
  "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"]
    }
  }
}
```

この場合、キー key_dと のみが、文字列または文字列の配列のいずれかである値が割り当てられているため、フィルターポリシーの一部としてカウントkey_fされます。親キー key_a、key_b、key_cにはネストされた JSON オブジェクトが値として含まれているため、カウントされません。

トピック

- 共通のポリシーの制約
- 属性ベースのフィルタリングのポリシー制約
- ペイロードベースのフィルタリングのポリシー制約

共通のポリシーの制約

- 文字列一致 – フィルターポリシーの文字列一致の場合、比較では大文字と小文字が区別されます。
- 数値マッチング – 数値マッチングの場合、値は -10^9 ~ 10^9 (-10 億 ~ 10 億) で、小数点以下 5 桁の精度になります。
- フィルターポリシーの複雑さ – フィルターポリシーの複雑さについて、値の合計が 150 を超えないようにする必要があります。合計の組み合わせを計算するには、フィルターポリシーの各配列の値の数を掛けます。

次のポリシーの例を考えてみましょう。

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

このポリシーでは、次のようになります。

- 最初の配列には 3 つの値があります
- 2 番目の配列には 1 つの値があります
- 3 番目の配列には 2 つの値があります

組み合わせの合計は次のように計算されます。

- 3 x 1 x 2 = 6

フィルターポリシー構文

フィルターポリシーの JSON には以下のものが含まれます。

- 引用符で囲まれた文字列
- 数字
- 引用符なしのキーワード `true`、`false`、および `null`

Amazon SNS API を使用する場合は、フィルターポリシーの JSON を有効な UTF-8 文字列として渡す必要があります。

フィルターポリシーの制限

- フィルターポリシーの最大サイズは 256 KB です。
- デフォルトでは、トピックごとに最大 200 のフィルターポリシー、 AWS アカウントごとに最大 10,000 のフィルターポリシーを設定できます。
- このポリシー制限により、 `Subscribe` API を使用した Amazon SQS キューサブスクリプションの作成が停止されることはありません。ただし、`Subscribe` API コール (または

SetSubscriptionAttributes API コール) にフィルターポリシーをアタッチすると失敗します。

- このクォータを引き上げるには、[AWS Service Quotas] を使用できます。

属性ベースのフィルタリングのポリシー制約

- 属性ベースのフィルタリングがデフォルトのオプションです。FilterPolicyScope がサブスクリプションでは MessageAttributes に設定されています。
- Amazon SNS は属性ベースのフィルタリングに対して、ネストされたフィルターポリシーを受け入れません。
- Amazon SNS は、以下のデータ型に該当するメッセージ属性とのみポリシーのプロパティを比較します。
  - String
  - String.Array

    > **Important**
    >
    > オブジェクトを配列で渡すことはお勧めしません。属性ベースのフィルタリングではサポートされていないネストが原因で、予期しない結果が発生する可能性があるためです。ネストされたポリシーにペイロードベースのフィルタリングを使用する

  - Number
- Amazon SNS は、Binary データ型のメッセージ属性を無視します。
- フィルターポリシーは、最大 5 個の属性名を持つことができます。

ペイロードベースのフィルタリングのポリシー制約

Amazon SNS は、ペイロードベースのフィルタリング用にネストされたフィルターポリシーを受け入れます。フィルターポリシーの値の合計を計算するために、ネストされた各配列の値の数を掛けます。

次のポリシーの例を考えてみましょう。

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

```
        }
    },
"key_d": {
    "key_e": ["value_one", "value_two", "value_three"]
    }
}
```

このポリシーでは、次のようになります。

- 最初の配列には、3 レベルのネストされたキーに 4 つの値があります。
- 2 番目の には、2 レベルのネストされたキーに 3 つの値があります。

組み合わせの合計は次のように計算されます。

- 4 x 3 x 3 x 2 = 72

ポリシーの制限

フィルターポリシーには、最大 5 つの親キー (最上位キー) を含めることができます。ネストされたポリシーの場合、親キーのみが 5 つのキー制限にカウントされます。

数値範囲

フィルターポリシーの数値マッチングの場合、値は -10^9 ~ 10^9 (-10 億 ~ 10 億) で、小数点以下 5 桁の精度になります。

ペイロードベースのフィルタリングへの切り替え

属性ベース (デフォルト) のフィルタリングからペイロードベースのフィルタリングに切り替えるには、サブスクリプションで `FilterPolicyScope` を `MessageBody` に設定する必要があります。

AND/OR ロジック

AND/OR ロジックを含むオペレーションを使用してメッセージ属性またはメッセージ本文と一致させることができます。

トピック

- AND ロジック
- OR ロジック
- OR 演算子

AND ロジック

複数のプロパティ名を使用して AND ロジックを適用できます。

次のポリシーについて考えます。

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

このポリシーは、customer_interests の値が rugby に設定され、さらに price_usd の値が 100 を超える値に設定されているメッセージ属性またはメッセージ本文プロパティと一致します。

> **Note**
>
> AND ロジックを同じ属性の値に適用することはできません。

OR ロジック

プロパティ名に複数の値を割り当てることで OR ロジックを適用できます。

次のポリシーについて考えます。

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

このポリシー属性は、customer_interests の値が rugby、football、または baseball に設定されているメッセージ属性またはメッセージ本文プロパティと一致します。

OR 演算子

"$or" 演算子を使用してフィルターポリシーを明示的に定義し、ポリシー内の複数の属性間の OR 関係を表すことができます。

Amazon SNS は、ポリシーが以下の条件をすべて満たしている場合にのみ "$or" 関係を認識します。これらの条件がすべて満たされない場合、"$or" はポリシー内の他の文字列と同様に通常の属性名として扱われます。

- ルールには "$or" フィールド属性が付けられ、その後に配列 (“$or” : [] など) が続きます。
- "$or" 配列には少なくとも次の 2 つのオブジェクトがあります: "$or": [{}, {}]。
- "$or" 配列内のどのオブジェクトにも、予約キーワードであるフィールド名はありません。

それ以外の場合は、ポリシー内の他の文字列と同様に、"$or" は通常の属性名として扱われます。

数値とプレフィックスは予約キーワードであるため、以下のポリシーは OR 関係として解析されません。

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

OR 演算子の例

標準の OR:

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

このポリシーのフィルターロジックは次のとおりです。

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

このポリシー属性は、以下のメッセージ属性セットのいずれとも一致します。

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

または

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

以下のメッセージ本文のいずれとも一致します。

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

または

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

OR 関係を含むポリシー制約

次のポリシーについて考えます。

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

このポリシーの口ジックは次のように簡略化することもできます。

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

OR 関係のあるポリシーの複雑さの計算は、各 OR ステートメントの組み合わせの複雑さの合計として簡略化できます。

組み合わせの合計は次のように計算されます。

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

source には 1 つの値、metricName には 2 つの値、metricType には 1 つの値、metricId には 2 つの値、spaceId には 3 つの値があります。

次のネストされたフィルターポリシーについて考えてみます。

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

このポリシーの口ジックは次のように簡略化できます。

```
("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")
```

組み合わせの合計の計算は、キーのネストレベルを考慮する必要がある点を除いて、ネストされていないポリシーでも同じです。

組み合わせの合計は次のように計算されます。

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

metricName には 2 つの値、namespace には 2 つの値があります。scope は、1 つの値を持つ 2 レベルのネストされたキー、source は 1 つの値を持つ 2 レベルのネストされたキー、type は 1 つの値を持つ 2 レベルのネストされたキーです。

キーの一致

exists 演算子では、フィルターポリシーで指定されたプロパティを使用して、または使用することなく、受信メッセージを一致できます。exists の一致は、リーフノードでのみ動作します。中間ノードでは機能しません。

- "exists": true を使用して、指定されたプロパティを含む受信メッセージと一致します。キーには、null でなく、空でない値が必要です。

  例えば、次のポリシーのプロパティは、true の値の exists 演算子を使用します。

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

  次のように、store 属性キーを含むすべてのメッセージ属性のリストに一致します。

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

  以下のメッセージ本文のいずれとも一致します。

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

  ただし、次のように、store 属性キーを持たないメッセージ属性の一覧とは一致しません。

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

  また、次のメッセージ本文にも一致しません。

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

```
}
```

- "exists": false を使用して、指定されたプロパティを含まない受信メッセージを一致します。

> **Note**
>
> "exists": false は少なくとも 1 つの属性が存在する場合にのみ一致します。属性のセットが空の場合、フィルターは一致しません。

例えば、次のポリシーのプロパティは、false の値の exists 演算子を使用します。

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

次のように、store 属性キーを含むすべてのメッセージ属性のリストに一致しません。

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

次のメッセージ本文にも一致しません。

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

ただし、次のように、store 属性キーを持たないメッセージ属性の一覧とは一致します。

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

以下のメッセージ本文にも一致します。

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

数値の一致

数値をメッセージ属性値またはメッセージ本文のプロパティ値に一致させることでメッセージをフィルタリングできます。JSON ポリシーでは、数値を二重引用符で囲みません。フィルターには次の数値オペレーションを使用できます。

> **Note**
>
> プレフィックスは、文字列一致のみがサポートされます。

トピック

完全一致

ポリシープロパティ値に numeric キーワードと演算子 = が含まれている場合、同じ名前と等しい数値を持つ任意のメッセージ属性またはメッセージ本文プロパティ値と一致します。

次のポリシーのプロパティについて考えます。

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

このポリシー属性は、以下のメッセージ属性のいずれとも一致します。

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

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

以下のメッセージ本文のいずれとも一致します。

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

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

「以外」のマッチング

ポリシープロパティ値にキーワード anything-but が含まれている場合、そのキーワードは、ポリシープロパティ値を含まない任意のメッセージ属性またはメッセージ本文プロパティ値と一致します。

次のポリシーのプロパティについて考えます。

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

このポリシー属性は、以下のメッセージ属性のいずれとも一致します。

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

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

以下のメッセージ本文のいずれとも一致します。

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

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

さらに、次のメッセージ属性にも一致します (これは 100 でも 500 でもない値を含むためです)。

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

また、次のメッセージ本文にも一致します (これは 100 でも 500 でもない値が含まれるためです)。

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

```
}
```

ただし、次のメッセージ属性とは一致しません。

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

また、次のメッセージ本文にも一致しません。

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

値範囲の一致

数値ポリシープロパティには、= 演算子に加えて、次の <、<=、>、および >= を含めることができます。

次のポリシーのプロパティについて考えます。

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

負の数値を持つ任意のメッセージ属性またはメッセージ本文プロパティと一致します。

別のポリシー属性について考えます。

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

最大 150 までの正の数を持つメッセージ属性またはメッセージ本文プロパティと一致します。

文字列値の一致

文字列値をメッセージ属性値またはメッセージ本文のプロパティ値に一致させることで、メッセージをフィルタリングできます。JSON ポリシーでは、文字列値を二重引用符で囲みます。次の文字列オペレーションを使用してメッセージ属性またはメッセージ本文と一致させることができます。

トピック

- 完全一致
- 「以外」のマッチング
- anything-but 演算子でプレフィックスを使用する。

- E quals-ignore-case マッチング
- IP アドレス一致
- プレフィックスマッチング
- サフィックスマッチング

完全一致

完全一致は、ポリシープロパティ値が 1 つ以上のメッセージ属性値と一致した場合に発生します。

次のポリシーのプロパティについて考えます。

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

このポリシー属性は、以下のメッセージ属性と一致します。

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

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

また、次のメッセージ本文にも一致します。

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

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

ただし、次のメッセージ属性とは一致しません。

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

また、次のメッセージ本文にも一致しません。

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

「以外」のマッチング

ポリシープロパティ値にキーワード anything-but が含まれている場合、そのキーワードは、ポリシープロパティ値を含まない任意のメッセージ属性またはメッセージ本文値と一致します。anything-but は、"exists": false と組み合わせることができます。

次のポリシーのプロパティについて考えます。

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

このポリシー属性は、以下のメッセージ属性のいずれとも一致します。

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

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

以下のメッセージ本文のいずれとも一致します。

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

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

さらに、次のメッセージ属性にも一致します (これは rugby でも tennis でもない値を含むためです)。

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

また、次のメッセージ本文にも一致します (これには rugby でも tennis でもない値が含まれるためです)。

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

ただし、次のメッセージ属性とは一致しません。

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

また、次のメッセージ本文にも一致しません。

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

anything-but 演算子でプレフィックスを使用する。

文字列一致では、anything-but 演算子と共にプレフィックスを使用することもできます。例えば、次のポリシープロパティが order- プレフィックスを拒否します。

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

このポリシー属性は、以下の属性のいずれとも一致します。

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

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

以下のメッセージ本文のいずれとも一致します。

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

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

ただし、次のメッセージ属性とは一致しません。

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

また、次のメッセージ本文にも一致しません。

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

```
}
```

E quals-ignore-case マッチング

ポリシープロパティに equals-ignore-case キーワードが含まれている場合、任意のメッセージ属性または本文プロパティ値を使用して、大文字と小文字を区別しない一致が実行されます。

次のポリシーのプロパティについて考えます。

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

このポリシー属性は、以下のメッセージ属性のいずれとも一致します。

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

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

以下のメッセージ本文のいずれとも一致します。

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

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

IP アドレス一致

cidr 演算子を使用して、受信メッセージが特定の IP アドレスまたはサブネットから発信されているかどうかを確認できます。

次のポリシーのプロパティについて考えます。

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

このポリシー属性は、以下のメッセージ属性のいずれとも一致します。

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

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

以下のメッセージ本文のいずれとも一致します。

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

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

ただし、次のメッセージ属性とは一致しません。

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

また、次のメッセージ本文にも一致しません。

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

プレフィックスマッチング

ポリシープロパティに prefix キーワードが含まれている場合、指定した文字で始まる任意のメッセージ属性または本文プロパティ値と一致します。

次のポリシーのプロパティについて考えます。

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

このポリシー属性は、以下のメッセージ属性のいずれとも一致します。

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

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

以下のメッセージ本文のいずれとも一致します。

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

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

ただし、次のメッセージ属性とは一致しません。

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

また、次のメッセージ本文にも一致しません。

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

サフィックスマッチング

ポリシープロパティに suffix キーワードが含まれている場合、指定した文字で終わる任意のメッセージ属性または本文プロパティ値と一致します。

次のポリシーのプロパティについて考えます。

```
"customer_interests": [{"suffix": "ball"}]
```

このポリシー属性は、以下のメッセージ属性のいずれとも一致します。

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

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

以下のメッセージ本文のいずれとも一致します。

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

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

ただし、次のメッセージ属性とは一致しません。

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

また、次のメッセージ本文にも一致しません。

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

サブスクリプションフィルターポリシーを適用する

Amazon SNS コンソールを使用して、Amazon SNS サブスクリプションにフィルターポリシーを適用できます。または、ポリシーをプログラム的に適用するには、Amazon SNS API、 AWS Command Line Interface (AWS CLI)、または Amazon SNS をサポートする任意の AWS SDK を使用できます。を使用することもできます。 AWS CloudFormation

> **Important**
>
> AWS IAM や Amazon SNS などのサービスは、結果整合性と呼ばれる分散コンピューティングモデルを使用します。サブスクリプションフィルターポリシーへの追加または変更は、完全に有効になるまでに最大 15 分かかります。

AWS Management Console

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[サブスクリプション] を選択します。
3. サブスクリプションを選択したら、[編集] を選択します。
4. [Edit](編集) ページで、[Subscription filter policy](サブスクリプションフィルターポリシーセクション) を展開します。
5. 属性ベースのフィルタリングまたはペイロードベースのフィルタリングのいずれかを選択します。

6. [JSON editor](JSON エディタ) フィールドで、フィルターポリシーの [JSON body](JSON 本文) を提供します。
7. [変更を保存] をクリックします。

   Amazon SNS により、フィルターポリシーがサブスクリプションに適用されます。

AWS CLI

AWS Command Line Interface (AWS CLI) を使用してフィルタポリシーを適用するには、set-subscription-attributes以下の例のようにコマンドを使用します。--attribute-name オプションで、FilterPolicy を指定します。--attribute-value で、JSON ポリシーを指定します。

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --
attribute-name FilterPolicy --attribute-value '{"store":["example_corp"],"event":
["order_placed"]}'
```

ポリシーに有効な JSON を提供するには、属性名と値を二重引用符で囲みます。また、ポリシーの引数全体を引用符で囲む必要があります。引用符のエスケープを避けるため、上に示した例のように、一重引用符を使用してポリシーを囲み、二重引用符を使用して JSON 名と値を囲みます。

属性ベース (デフォルト) からペイロードベースのメッセージフィルタリングに切り替える場合は、コマンドを使用することもできます。set-subscription-attributes--attribute-name オプションで、FilterPolicyScope を指定します。--attribute-value の場合、MessageBody を指定します。

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-
name FilterPolicyScope --attribute-value MessageBody
```

フィルターポリシーが適用されたことを確認するには、get-subscription-attributes コマンドを使用します。ターミナル出力の属性には、以下の例に示すように、FilterPolicy キーのフィルターポリシーが表示されます。

```
$ 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 SDK

SetSubscriptionAttributes次のコード例は使用方法を示しています。

> **Important**
>
> SDK for Java 2.x の例を使用している場合、クラス SNSMessageFilterPolicy はそのままでは使用できません。このクラスのインストール方法については、 GitHub Web サイトの例を参照してください。

CLI

AWS CLI

サブスクリプション属性を設定するには

次の set-subscription-attributes の例では、RawMessageDelivery 属性を SQS サブスクリプションに設定します。

```
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
```

このコマンドでは何も出力されません。

次の set-subscription-attributes の例では、FilterPolicy 属性を SQS サブスクリプションに設定します。

```
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\"] }"
```

このコマンドでは何も出力されません。

次の `set-subscription-attributes` の例では、`FilterPolicy` 属性を SQS サブスクリプションから削除します。

```
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 "{}"
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」SetSubscriptionAttributesのを参照してください。

Java

SDK for Java 2.x

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API SetSubscriptionAttributesリファレンスのを参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスのを参照してください[SetSubscriptionAttributes](#)。

Amazon SNS API

Amazon SNS API を使用してフィルターポリシーを適用するには、[SetSubscriptionAttributes](#) アクションへのリクエストを作成します。`AttributeName` パラメータを `FilterPolicy` に設定し、`AttributeValue` パラメータをフィルターポリシーの JSON に設定します。

メッセージフィルタリングを属性ベース (デフォルト) からペイロードベースのメッセージフィルタリングに切り替える場合は、`SetSubscriptionAttributes` アクションも使用できます。`AttributeName` パラメータを `FilterPolicyScope` に設定し、`AttributeValue` パラメータを `MessageBody` に設定します。

AWS CloudFormation

を使用してフィルターポリシーを適用するには AWS CloudFormation、JSON または YAML テンプレートを使用してスタックを作成します。 AWS CloudFormation 詳細については、『AWS CloudFormation ユーザーガイド』*`FilterPolicyAWS::SNS::Subscription`* AWS CloudFormation のリソースのプロパティとサンプルテンプレートを参照してください。

1. AWS CloudFormation コンソール にサインインします。
2. [スタックの作成] を選択します。
3. [テンプレートの選択] ページで、[テンプレートを Amazon S3 にアップロード] を選択してから、ファイルを選択して [次へ] をクリックします。
4. [詳細の指定] ページで、以下の作業を行います。
   a. [スタックの名前] に「`MyFilterPolicyStack`」と入力します。
   b. にはmyHttpEndpoint、トピックに登録する HTTP エンドポイントを入力します。

   > **Tip**
   > HTTP エンドポイントがない場合は作成します。

5. [オプション] ページで、[次へ] をクリックします。
6. [Review] ページで、[作成] を選択します。

サブスクリプションフィルターポリシーを削除する

サブスクリプションに送信されるメッセージのフィルター処理を停止するには、サブスクリプションのフィルターポリシーを空の JSON 本文に上書きすることで削除します。このポリシーを削除したら、サブスクリプションは発行されるすべてのメッセージを受け取るようになります。

AWS Management Console

1. Amazon SNS コンソールにサインインします。

2. ナビゲーションパネルで、[サブスクリプション] を選択します。
3. サブスクリプションを選択したら、[編集] を選択します。
4. [Edit ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***] ページで、[サブスクリプションフィルターポリシー] セクションを展開します。
5. [JSON エディタ] フィールドで、フィルターポリシーの空の JSON 本文を指定します ({})。
6. [変更を保存] をクリックします。

   Amazon SNS により、フィルターポリシーがサブスクリプションに適用されます。

AWS CLI

AWS CLI を使用してフィルターポリシーを削除するには、`set-subscription-attributes` コマンドを使用し、`--attribute-value` 引数に空の JSON 本文を指定します。

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-name FilterPolicy --attribute-value "{}"
```

Amazon SNS API

Amazon SNS API を使用してフィルターポリシーを削除するには、`SetSubscriptionAttributes` アクションへのリクエストを行います。`AttributeName` パラメータを `FilterPolicy` に設定し、`AttributeValue` パラメータに空の JSON 本文を指定します。

メッセージデータ保護

トピック

メッセージデータ保護とは

メッセージデータ保護は、データ保護ポリシーアプリケーションまたは AWS サービス間を移動する機密情報を監査、マスク、編集またはブロックすることにより、Amazon SNS トピックに公開されるデータを保護します。

メッセージデータ保護は、データ識別子を使って、個人を特定できる情報 (PII) および保護医療情報 (PHI) に関する移動中のデータをスキャンします。事前定義された (または Amazon SNS マネージド) データ識別子 (名前、住所、クレジットカード番号、処方薬コードなど) の使用を選択するか、ビジネスユースケースに固有の独自のカスタムデータ識別子を作成できます。メッセージデータ保護は、スキャンした情報を使用して詳細な監査ログを提供するため、データを保護するための措置を講じることができます。

メッセージデータ保護は、機密性の高い顧客情報を保護するために以下のアクションをサポートしています。

- 監査 — Amazon SNS トピックに公開されたデータの最大 99% を監査します。その後、検出結果を Amazon CloudWatch、Amazon S3、または Amazon Data Firehose に送信することを選択できます。
- 識別解除 — メッセージの発行や配信を中断することなく、機密データをマスクまたは削除できます。
- 拒否 — 機密データがペイロード内に存在する場合、アプリケーションと AWS リソース間のデータ送信をブロックします。

> **Note**
>
> Amazon SNS は Amazon SNS 標準トピックのみのメッセージデータ保護をサポートします。

メッセージデータ保護を使用すべき理由

ガバナンス、リスク管理、コンプライアンスプログラムにメッセージデータ保護を導入することで、データ漏洩の特定と防止に役立つデータ保護ポリシーを実装できます。これにより、HIPAA、GDPR、PCI、FedRAMP などのプライバシー規制に準拠することで、財務、法律、規制上のリスクを軽減するのに役立つツールをチームに提供できます。また、開発者は、機密データを保護するための独自のツールの構築および管理に伴う運用上のオーバーヘッドから解放されます。

たとえば、メッセージデータ保護を使用して監査ポリシーを作成し、誤って機密データを送受信しているシステムがないかどうかを確認できます。監査結果から、システムがクレジットカード情報を必要としないシステムに送信していることが明らかになった場合は、ブロックポリシーを使用してデータの配信を防止できます。

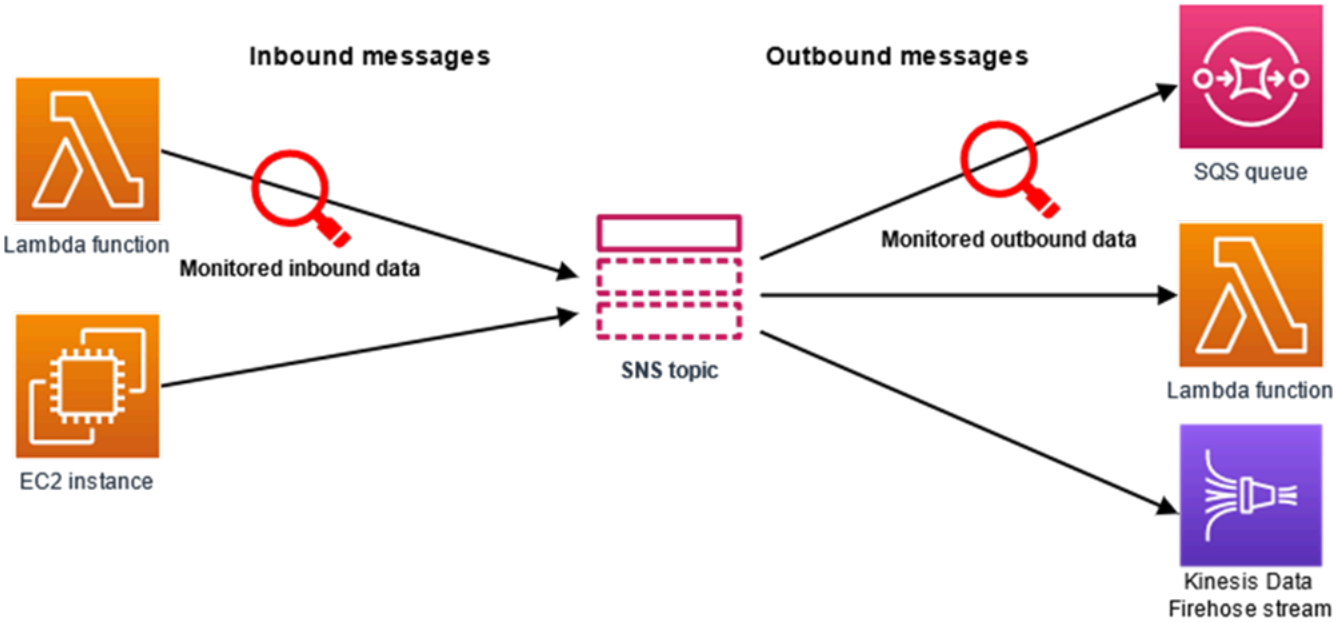
> **Note**
>
> Amazon SNS は Amazon SNS 標準トピックのみのメッセージデータ保護をサポートします。

データ保護ポリシーを理解する

トピック

データ保護ポリシーとは

Amazon SNS はデータ保護ポリシーを使って、スキャンする機密データと、そのデータが Amazon SNS トピックで交換されないように保護するアクションを選択します。目的の機密データを選択するには、データ識別子を使用します。次に、Amazon SNS メッセージデータ保護は、機械学習とパターンマッチングを使用して機密データを検出します。見つかったデータ識別子に基づいて、監査、識別解除、または拒否のオペレーションを定義できます。これらのオペレーションにより、見つかった (または見つからなかった) 機密データを口グに記録、機密データをマスクまたは編集、またはメッセージの配信を拒否できます。

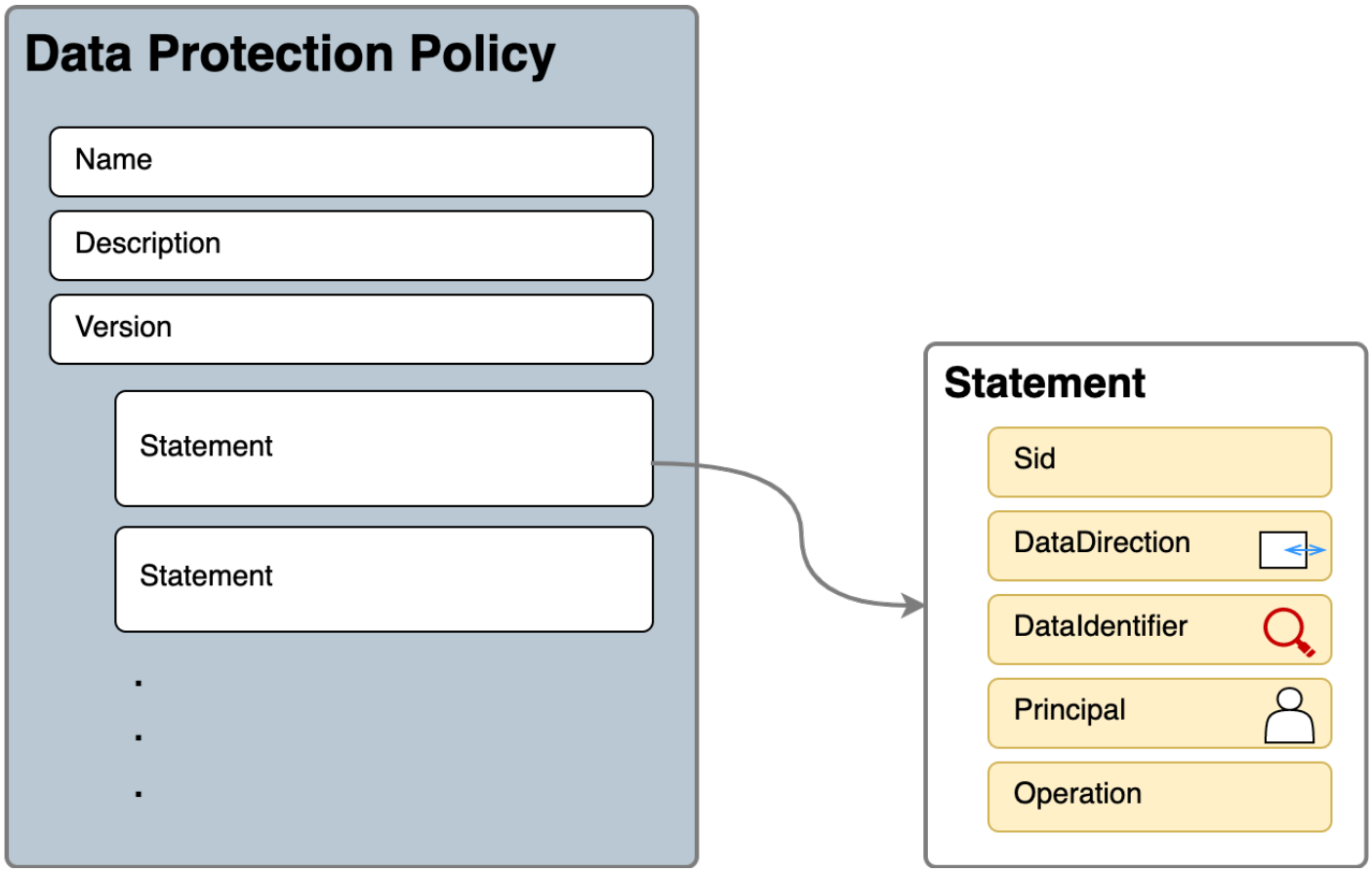


データ保護ポリシーの構成の仕組み

次の図に示すように、データ保護ポリシードキュメントには次の要素が含まれています。

- ドキュメントの最上部に記載されるポリシー全体の情報 (任意)
- 1 つ以上の個別のステートメント

各ステートメントには、1 つのアクセス許可に関する情報が含まれています。



Amazon SNS トピックごとに定義できるデータ保護ポリシーは 1 つだけです。データ保護ポリシーには、1 つまたは複数の拒否または識別解除ステートメントと 1 つの監査ステートメントのみを含めることができます。

データ保護ポリシーの JSON プロパティ

データ保護ポリシーでは、識別のために以下の基本ポリシー情報が必要です。

- Name – ポリシーの名前。
- Description (オプション) – ポリシーの説明。
- Version – ポリシー言語のバージョン。現在のバージョンは 2021-06-01. です。
- Statement – データ保護ポリシーアクションを指定するステートメントのリスト。

```
{
  "Name": "basicPII-protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
```

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

ポリシーステートメントの JSON プロパティ

ポリシーステートメントは、データ保護オペレーションの検出コンテキストを設定します。

- [Sid] (オプション) — ステートメントの識別子。
- [DataDirection] (データ方向) — Amazon SNS トピックに関するインバウンド (API リクエストの発行) またはアウトバウンド (通知配信)。
- [DataIdentifier] (データ識別子) — Amazon SNS トピックがスキャンすべき機密データ。名前、住所、電話番号などです。
- [プリンシパル] — トピックを発行した IAM プリンシパル、またはトピックにサブスクライブされた IAM プリンシパル。
- [Operation] (オペレーション) — 機密データが見つかると Amazon SNS トピックが実行する、[Audit] (監査)、[De-identify] (マスクまたは編集)、または [Deny] (拒否) (ブロック) のいずれかである後続のアクション。

```
{
    "Sid": "basicPII-inbound-protection",
    "DataDirection": "Inbound",
    "Principal": ["*"],
    "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Name",
        "arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US"
    ],
    "Operation": {
        ...
    }
}
```

ポリシーステートメントオペレーションの JSON プロパティ

ポリシーステートメントは、次のデータ保護オペレーションのいずれかを設定します。

- [Audit] (監査) — メッセージの発行や配信を中断することなく、メトリクスを出力してログを検索します。

- 識別解除 — メッセージの公開を中断することなく、機密データをマスク、または編集できます。
- [Deny] (拒否) — Amazon SNS 発行リクエストをブロックするか、メッセージの配信に失敗します。

データ保護ポリシーの IAM プリンシパルを決定する方法

メッセージデータ保護では、Amazon SNS とやり取りする 2 つの IAM プリンシパルを使用します。

1. [Publish API Principal] (API プリンシパルの発行) (インバウンド) — Amazon SNS `Publish` API を呼び出す認証済みの IAM プリンシパル。
2. [Subscription Principal] (サブスクリプションプリンシパル) (アウトバウンド) — サブスクリプションの作成中に `Subscribe` API を呼び出した認証済みの IAM プリンシパル。

`SubscriptionPrincipal` は公開されている Amazon SNS サブスクリプションプロパティで、`GetSubscriptionAttributes` API から取得できます。

```
{
  "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"
  }
}
```

データ保護ポリシーオペレーション

以下は、機密データの監査および拒否に使用できるデータ保護ポリシーの例です。サンプルアプリケーションを含む完全なチュートリアルについては、「Introducing message data protection for Amazon SNS」(Amazon SNS のメッセージデータ保護の実装) のブログ記事を参照してください。

トピック

- 監査オペレーション
- 識別解除オペレーション
- 拒否オペレーション

監査オペレーション

[Audit] (監査) オペレーションは、トピックのインバウンドメッセージをサンプリングし、機密データの結果を AWS の送信先に口ギングします。サンプルレートは、0 ~ 99 の整数になります。このオペレーションには、次のいずれかのタイプの口ギング先が必要です。

1. FindingsDestination – Amazon SNS トピックがペイロード内の機密データを検出したときの口グ記録の送信先。
2. NoFindingsDestination – Amazon SNS トピックがペイロードで機密データを見つけられない場合の口グ記録先。

次の AWS のサービス を以下の各口グの送信先タイプで使用できます。

- Amazon CloudWatch Logs (オプション) – LogGroupはトピックリージョンにあり、名前は /aws/vendedlogs/ で始まる必要があります。
- Amazon Data Firehose （ オプション) – はトピックリージョンにあり、配信ストリームのソースとしてダイレクト PUT を持っているDeliveryStream必要があります。詳細については、「Amazon Data Firehose デベロッパーガイド」の「送信元、送信先、名前」を参照してください。
- Amazon S3 (オプション) — Amazon S3 バケット名。SSE-KMS 暗号化を有効にして Amazon S3 バケットを使用するには、追加のアクションが必要です。

```
{
  "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"
            }
        }
    }
  }
}
```

ログの送信先の指定に必要な権限

データ保護ポリシーで口ギング先を指定する場合、Amazon SNS `PutDataProtectionPolicy` API、または `--data-protection-policy` パラメータを含む `CreateTopic` API を呼び出す IAM プリンシパルの IAM ID ポリシーに、次のアクセス権限を追加する必要があります。

監査先	IAM 許可
デフォルト値	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups

監査先	IAM 許可
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy SSE-KMS 暗号化を有効にして Amazon S3 バケットを使用するには、追加のアクションが必要です。

```
{
  "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"
      ]
    }
  ]
}
```

SSE-KMS に使用する必須のキーポリシー

Amazon S3 バケットを口グの送信先として使用する場合、バケット内のデータを保護するには、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) または AWS KMS keys によるサーバー側の暗号化 (SSE-KMS) のいずれかを有効にします。詳細については、Amazon S3 ユーザーガイドの「サーバー側の暗号化を使用したデータの保護」をご参照ください。

SSE-S3 を選択した場合、追加の設定は必要ありません。Amazon S3 が暗号化キーを処理します。

SSE-KMS を選択した場合は、カスタマーマネージドキーを使用する必要があります。口グ配信アカウントが S3 バケットに書き込めるように、カスタマーマネージドキーのキーポリシーを更新する必要があります。SSE-KMS で使用するために必要なキーポリシーの詳細については、「Amazon Logs ユーザーガイド」の「Amazon S3 バケットのサーバー側の暗号化 CloudWatch 」を参照してください。

監査先口グの例

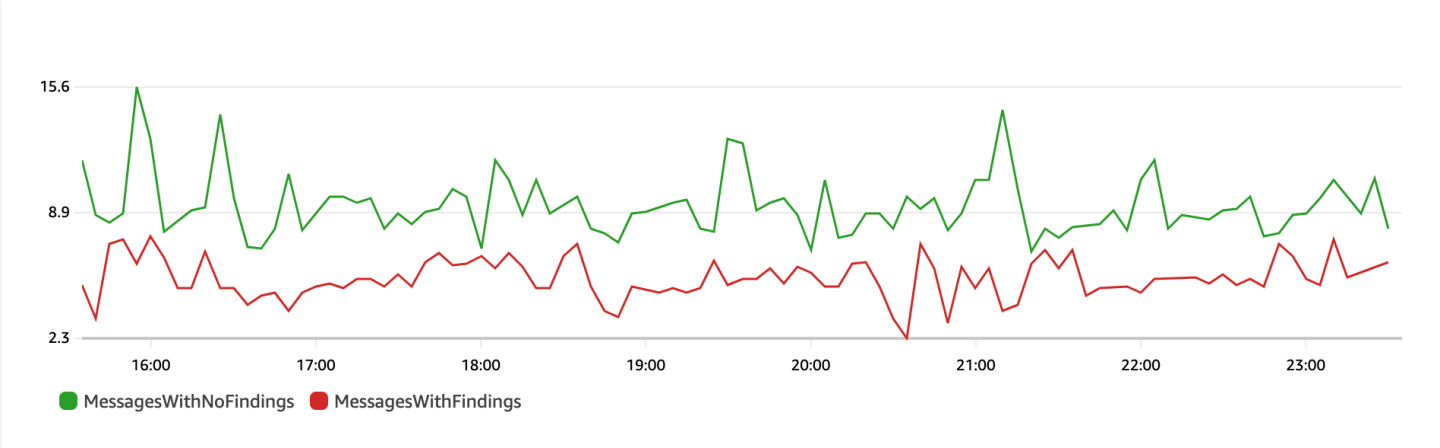
次の例では、`callerPrincipal` を使用して機密コンテンツのソースを特定します。また、`messageID` を参照として使用し、`Publish` API レスポンスに対してチェックします。

```
{
```

```
  "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
        }
      ]
    }
  ]
}
```

監査オペレーションのメトリクス

監査オペレーションが FindingsDestinationまたは NoFindingsDestinationプロパティを指定すると、トピック所有者も MessagesWithFindingsおよび MessagesWithNoFindingsメトリクスを受け取ります CloudWatch。



識別解除オペレーション

識別解除オペレーションは、発行または配信するメッセージの機密データをマスキングまたは削除します。このオペレーションは、インバウンドとアウトバウンドの両方のメッセージに使用可能であり、次のいずれかのタイプの設定を必要とします。

- MaskConfig – 次の表でサポートされている文字を使用してマスクします。例えば、ssn: 123-45-6789 は ssn: ########### になります。

```
{
"Operation": {
    "Deidentify": {
        "MaskConfig": {
            "MaskWithCharacter": "#"
          }
    }
}
```

サポートされているマスク文字	名前
*	アスタリスク
A ~ Z、a ~ z、0 ~ 9	英数字
	スペース
!	感嘆符

サポートされているマスク文字	名前
$	ドル記号
%	パーセント記号
&	アンパサンド
()	かっこ
+	プラス記号
,	カンマ
-	ハイフン
.	[Period] (期間)
/\	スラッシュ、バックスラッシュ
#	ナンバー記号
:	コロン
;	セミコロン
=, <>	等号。小なりまたは大なり
@	アットマーク
[]	角かっこ
^	キャレット記号
_	下線
`	バックティック
|	縦棒
~	チルダ記号

- RedactConfig - データを完全に削除して編集します。例えば、ssn: 123-45-6789 は ssn:　になります。

```
{
"Operation": {
    "Deidentify": {
        "RedactConfig": {}
    }
}
```

受信メッセージでは、監査オペレーション後に機密データが識別解除され、メッセージ全体が機密である場合、SNS:Publish API 発信者は次の無効なパラメータエラーを受け取ります。

```
Error code: AuthorizationError ...
```

拒否オペレーション

[Deny] (拒否) オペレーションは、メッセージに機密データが含まれている場合に Publish API リクエストまたはメッセージの配信のいずれかを中断します。拒否オペレーションオブジェクトは、追加の設定を必要としないため空です。

```
"Operation": {
    "Deny": {}
}
```

インバウンドメッセージでは、SNS:Publish API 発信者が認証エラーを受け取ります。

```
Error code: AuthorizationError ...
```

アウトバウンドメッセージでは、Amazon SNS トピックはメッセージをサブスクリプションに配信しません。不正配信を追跡するには、トピックの[delivery status logging] (配信ステータスのログ記録) を有効にします。以下は、配信ステータスログの例です。

```
{
    "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"
}
```

データ保護ポリシーの例

以下は、機密データの監査および拒否に使用できるデータ保護ポリシーの例です。サンプルアプリケーションを含む完全なチュートリアルについては、「Introducing message data protection for Amazon SNS」(Amazon SNS のメッセージデータ保護の実装) のブログ記事を参照してください。

トピック

- 監査ポリシーの例
- インバウンド識別解除マスキングステートメントを含むポリシーの例
- インバウンド識別解除の削除ステートメントを使用したポリシーの例
- アウトバウンド識別解除マスキングステートメントを使用したポリシーの例
- アウトバウンド識別解除の削除ステートメントを使用したポリシーの例
- インバウンド拒否ステートメントを使用するポリシーの例
- アウトバウンド拒否ステートメントを使用するポリシーの例

監査ポリシーの例

監査ポリシーを使用すると、インバウンドメッセージの最大 99% を監査し、結果を Amazon CloudWatch、Amazon Data Firehose 、および Amazon S3 に送信できます。

例えば、監査ポリシーを作成して、システムのいずれかが誤って機密データを送受信していないかを評価できます。監査結果から、システムがクレジットカード情報を必要としないシステムにその情報を送信していることが明らかになった場合は、データ保護ポリシーを実装してデータの配信を防止できます。

次の例では、クレジットカード番号を探し、結果を CloudWatch Logs、Firehose、Amazon S3 に送信することで、トピックを通過するメッセージの 99% を監査します。

データ保護ポリシー:

```
{
  "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>"
            }
          }
        }
      }
    }
  ]
}
```

監査結果の形式の例:

```
{
    "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"
}
```

インバウンド識別解除マスキングステートメントを含むポリシーの例

次の例では、メッセージの内容から機密データをマスキングして、ユーザーがメッセージ内の CreditCardNumber をトピックに発行できないようにします。

```
{
  "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": "#"
          }
        }
      }
    }
  ]
}
```

インバウンド識別解除マスキング結果の例:

```
// original message
My credit card number is 4539894458086459
```

```
// delivered message
My credit card number is ################
```

インバウンド識別解除の削除ステートメントを使用したポリシーの例

次の例では、メッセージ内の機密データを削除して、ユーザーがメッセージ内の CreditCardNumber をトピックに発行できないようにします。

```
{
  "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": {}
        }
      }
    }
  ]
}
```

インバウンド識別解除の削除結果の例:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

アウトバウンド識別解除マスキングステートメントを使用したポリシーの例

次の例では、メッセージ内の機密データをマスキングして、ユーザーがメッセージ内の CreditCardNumber を受信できないようにします。

```
{
  "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": "-"
          }
        }
      }
    }
  ]
}
```

アウトバウンド識別解除マスキング結果の例:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ----------------
```

アウトバウンド識別解除の削除ステートメントを使用したポリシーの例

次の例では、メッセージ内の機密データを削除して、ユーザーがメッセージ内の CreditCardNumber を受信できないようにします。

```
{
  "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": {}
        }
      }
    }
  ]
}
```

アウトバウンド識別解除の削除結果の例:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

インバウンド拒否ステートメントを使用するポリシーの例

次の例では、メッセージの内容に CreditCardNumber が含まれている場合、ユーザーがメッセージをトピックに発行するのをブロックします。API レスポンスで拒否されたペイロードのステータスコードは「403 AuthorizationError」です。

```
{
  "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": {}
      }
    }
  ]
}
```

アウトバウンド拒否ステートメントを使用するポリシーの例

次の例では、AWS アカウントが CreditCardNumber を含むメッセージを受信しないようにブロックします。

```
{
  "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": {}
      }
    }
  ]
}
```

Amazon に記録されたアウトバウンド拒否結果の例 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"
}
```

データ保護ポリシーの作成

データ保護ポリシーは、アプリケーションまたは AWS のサービス 間を移動する機密情報を監査、識別解除 (マスキングまたは編集)、および拒否 (ブロック) することで、Amazon SNS トピックに対して発行されたデータを保護します。AWSAPI、AWS CLI、AWS CloudFormation、または AWS Management Console を使ってデータ保護ポリシーを作成できます。Amazon SNS トピックごとに定義できるポリシーは 1 つだけです。各データ保護ポリシーには、1 つまたは複数の識別解除および拒否ステートメントと 1 つの監査ステートメントのみを含めることができます。

トピック

- メッセージデータ (API) を保護するためのデータ保護ポリシーの作成
- メッセージデータ (CLI) を保護するためのデータ保護ポリシーの作成
- メッセージデータを保護するためのデータ保護ポリシーの作成 (CloudFormation)
- メッセージデータ (コンソール) を保護するためのデータ保護ポリシーの作成
- メッセージデータを保護するためのデータ保護ポリシーの作成 (SDK)

メッセージデータ (API) を保護するためのデータ保護ポリシーの作成

AWS アカウントの Amazon SNS リソースの数とサイズには制限があります。詳細については、「Amazon Simple Notification Service のエンドポイントとクォータ」を参照してください。

データ保護ポリシーの作成 (AWS API)

AWS API を使って Amazon SNS データ保護ポリシーを作成することができます。

Amazon SNS トピック (AWS API) と一緒にデータ保護ポリシーを作成するには

標準 Amazon SNS トピックの `DataProtectionPolicy` プロパティの使用 :

- `CreateTopic`

既存の Amazon SNS トピック (AWS API) のデータ保護ポリシーを作成または取得するには

以下のいずれかのオペレーションを呼び出します。

- GetDataProtectionPolicy
- PutDataProtectionPolicy

メッセージデータ (CLI) を保護するためのデータ保護ポリシーの作成

AWS アカウントの Amazon SNS リソースの数とサイズには制限があります。詳細については、「Amazon Simple Notification Service のエンドポイントとクォータ」を参照してください。

データ保護ポリシーの作成 (AWS CLI)

AWS Command Line Interface を使って Amazon SNS データ保護ポリシーを作成することができます。

Amazon SNS トピック (AWS CLI) と一緒にデータ保護ポリシーを作成するには

このオプションを使用して、標準の Amazon SNS トピックと一緒に新しいデータ保護ポリシーを作成します。

- create-topic

既存の Amazon SNS トピック (AWS CLI) のデータ保護ポリシーを作成または取得するには

以下のいずれかのオペレーションを呼び出します。

- get-data-protection-policy
- put-data-protection-policy

メッセージデータを保護するためのデータ保護ポリシーの作成 (CloudFormation)

AWS アカウントの Amazon SNS リソースの数とサイズには制限があります。詳細については、「Amazon Simple Notification Service のエンドポイントとクォータ」を参照してください。

データ保護ポリシーの作成 (CloudFormation)

AWS CloudFormation を使って Amazon SNS データ保護ポリシーを作成することができます。

Amazon SNS トピック (CloudFormation) と一緒にデータ保護ポリシーを作成するには

このオプションを使用して、標準の Amazon SNS トピックと一緒に新しいデータ保護ポリシーを作成します。

- AWS::SNS::Topic

メッセージデータ (コンソール) を保護するためのデータ保護ポリシーの作成

AWS アカウントの Amazon SNS リソースの数とサイズには制限があります。詳細については、「Amazon Simple Notification Service のエンドポイントとクォータ」を参照してください。

Amazon SNS トピックとともにデータ保護ポリシーを作成するには (コンソール)

このオプションを使用して、標準の Amazon SNS トピックと一緒に新しいデータ保護ポリシーを作成します。

1. Amazon SNS コンソールにサインインします。
2. トピックを選択するか、新しいトピックを作成できます。トピックの作成の詳細については、「Amazon SNS のトピックの作成」を参照してください。
3. [Create topic] (トピックの作成) ページの [Details] (詳細) セクションで [Standard] (標準) を選択します。

   a. トピックの名前を入力します。

   b. (オプション) トピックの表示名を入力します。
4. [Data protection policy] (データ保護ポリシー) を展開します。
5. [Configuration mode] (設定モード) を選択します。

   - [Basic] (ベーシック) — シンプルなメニューを使用してデータ保護ポリシーを定義します。

   - [Advanced] (アドバンスト) — JSON を使用してカスタムデータ保護ポリシーを定義します。
6. (オプション) 独自のカスタムデータ識別子を作成するには、[カスタムデータ識別子の設定セクション] を展開し、次の操作を行います。

a. カスタムデータ識別子の一意の名前を入力します。カスタムデータ識別子名には、英数字、アンダースコア (_)、ハイフン (-) を使用できます。カスタム識別子名の長さは最大 128 文字です。カスタムデータ識別子は、マネージドデータ識別子と同じ名前を共有することはできません。カスタムデータ識別子の制限の詳細な一覧については、「カスタムデータ識別子の制約」を参照してください。

b. カスタムデータ識別子に正規表現 (RegEx) を入力します。 RegEx は、英数字、 RegEx 予約文字、記号をサポートしています。 RegEx の最大長は 200 文字です。 RegEx が複雑すぎると、Amazon SNS は API コールに失敗します。制限事項の完全な RegExリストについては、「」を参照してくださいカスタムデータ識別子の制約。

c. (オプション) 必要に応じて [カスタムデータ識別子の追加] を選択し、データ識別子を追加します。各データ保護ポリシーに使用できるカスタムデータ識別子は最大 10個です。

7. データ保護ポリシーに追加したいステートメントを選択します。監査、識別解除 (マスクまたは編集)、拒否 (ブロック) の各ステートメントタイプを同じデータ保護ポリシーに追加できます。

a. [Add audit statement] (監査ステートメントの追加) — 監査する機密データ、そのデータについて監査するメッセージの割合、監査ログの送信先を設定します。

> **Note**
>
> データ保護ポリシーまたはトピックごとに許可される監査ステートメントは 1 つだけです。

i. データ識別子を指定して監査する機密データを定義します。

ii. [Audit sample rate] (監査サンプルレート) に、機密情報を監査するメッセージの割合を最大 99% で入力します。

iii. [Audit destination] (監査先) では、どの AWS のサービス に監査結果を送信するか、使用する各 AWS のサービス の送信先名を入力します。次のAmazon Web Services から選択できます。

- Amazon CloudWatch - CloudWatch Logs はAWS標準のログ記録ソリューションです。 CloudWatch Logs を使用すると、Logs Insights を使用してログ分析を実行し (ここでのサンプルを参照) 、メトリクスとアラームを作成できます。 CloudWatch Logs は、多くの のサービスがログを発行する場所であるため、1 つのソリューションを使用してすべてのログを簡単に集約できます。Amazon の詳細については CloudWatch、「Amazon CloudWatch ユーザーガイド」を参照してください。

- Amazon Data Firehose – Firehose は、Splunk へのリアルタイムストリーミングの需要を満たし OpenSearch、Amazon Redshift はさらにログ分析を行います。Amazon Data Firehose の詳細については、「Amazon Data Firehose ユーザーガイド」を参照してください。
- Amazon Simple Storage Service — Amazon S3 は、アーカイブ用のログ記録先として経済的です。ログは、数年間にわたって保持が必要な場合があります。こうした場合、ログを Amazon S3 に保存することで、コストを節約できます。Amazon Simple Storage Service の詳細については、「Amazon Simple Storage Service ユーザーガイド」を参照してください。

b. 識別解除ステートメントの追加 — メッセージで識別解除したい機密データを設定して、そのデータをマスクまたは編集するか、アカウントでそのデータの配信を停止します。

i. データ識別子の場合は、識別解除する機密データを選択します。

ii. [Define this de-identify statement for] (次についてのこの識別解除ステートメントを定義:) で、この識別解除ステートメントが適用される AWS アカウントまたは IAM プリンシパルを選択します。これはすべての AWS アカウント、または特定の AWS アカウントまたはアカウント ID または IAM エンティティ ARN を使用する IAM エンティティ (アカウントルート、ロール、またはユーザー) に適用できます。複数の ID または ARN を設定するには、それぞれをカンマ (,) で区切ります。

サポートされている IAM プリンシパルは次のとおりです。

- IAM アカウントプリンシパル — 例: `arn:aws:iam::AWS-account-ID:root`。
- IAM ロールプリンシパル — 例: `arn:aws:iam::AWS-account-ID:role/role-name`。
- IAM ユーザープリンシパル — 例: `arn:aws:iam::AWS-account-ID:user/user-name`。

iii. 識別解除オプションの場合は、機密データを識別解除する方法を選択します。以下のオペレーションがサポートされています。

- 編集 — データを完全に削除します。例えば、email: `classified@amazon.com` は email: になります。
- マスク — データを単一の文字に置き換えます。例えば、email: `classified@amazon.com` は email: `**********` になります。

iv. (オプション) 必要に応じて、さらに識別解除ステートメントを追加します。

c. [Add deny statement] (拒否ステートメントの追加) — トピック内を移動しないようにする機密データと、そのデータの送信を防ぐプリンシパルを構成します。

   i. データ方向については、拒否ステートメントのメッセージの方向を選択します。

      - [Inbound messages] (インバウンドメッセージ) — この拒否ステートメントをトピックに送信されるメッセージに適用します。
      - [Outbound messages] (アウトバウンドメッセージ) — この拒否ステートメントを、トピックがサブスクリプションエンドポイントに配信するメッセージに適用します。

   ii. データ識別子を選択して拒否する機密データを定義します。

   iii. この拒否ステートメントに適用する IAM プリンシパルを選択します。すべての AWS アカウント、または特定の AWS アカウント、またはアカウント ID または IAM エンティティ ARN を使用する IAM エンティティ (例: アカウントルート、ロール、またはユーザー) に適用できます。複数の ID または ARN を設定するには、それぞれをカンマ (,) で区切ります。サポートされている IAM プリンシパルは次のとおりです。

      - IAM アカウントプリンシパル — 例: `arn:aws:iam::AWS-account-ID:root`。
      - IAM ロールプリンシパル — 例: `arn:aws:iam::AWS-account-ID:role/role-name`。
      - IAM ユーザープリンシパル — 例: `arn:aws:iam::AWS-account-ID:user/user-name`。

   iv. (オプション) 必要に応じて、さらに拒否ステートメントを追加します。

メッセージデータを保護するためのデータ保護ポリシーの作成 (SDK)

AWS アカウントの Amazon SNS リソースの数とサイズには制限があります。詳細については、「Amazon Simple Notification Service のエンドポイントとクォータ」を参照してください。

データ保護ポリシーの作成 (AWS SDK)

AWS SDK を使って Amazon SNS データ保護ポリシーを作成することができます。

Amazon SNS トピックと一緒にデータ保護ポリシーを作成するには (AWS SDK)

以下のオプションを使用して、標準の Amazon SNS トピックと一緒に新しいデータ保護ポリシーを作成します。

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();
```

既存の Amazon SNS トピックのデータ保護ポリシーを作成または取得するには (AWS SDK)

以下のオプションを使用して、標準の Amazon SNS トピックと一緒に新しいデータ保護ポリシーを作成または取得します。

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();
```

Amazon SNS でデータ保護ポリシーを削除する

AWS API、AWS CLI、AWS CloudFormation、または AWS Management Console を使用して、Amazon SNS データ保護ポリシーを削除できます。

Amazon SNS データ保護ポリシーの一般情報については、「データ保護ポリシーを理解する」を参照してください。

AWS アカウントの Amazon SNS データ保護ポリシーリソースの数とサイズには制限があります。詳細については、「AWS 全般のリファレンス」の「Amazon SNS API スロットリング」を参照してください。

トピック

- データ保護ポリシーの削除 (コンソール)
- 空の JSON 文字列を使用してデータ保護ポリシーを削除する
- AWS CLI を使用してデータ保護ポリシーを削除する

データ保護ポリシーの削除 (コンソール)

マネージドデータ保護ポリシーを削除するには (コンソール)

1. Amazon SNS コンソールにサインインします。
2. 削除するデータ保護ポリシーを含むトピックを選択します。
3. [Edit] (編集) を選択します。
4. [Data protection policy] (データ保護ポリシー) セクションを展開します。
5. 削除するデータ保護ポリシーステートメントの横にある [Remove] (削除) を選択します。
6. [Save changes] (変更の保存) をクリックします。

空の JSON 文字列を使用してデータ保護ポリシーを削除する

データ保護ポリシーは、空の JSON 文字列に更新することで削除できます。

AWS CLI を使用してデータ保護ポリシーを削除する

AWS CLI を使用してデータ保護ポリシーを削除できます。

```
//aws sns put-data-protection-policy --resource-arn topic-arn --data-
protection-policy ""
```

データ識別子

Amazon SNS は、機械学習やパターンマッチングなどの基準と手法の組み合わせを使用して、機密データを検出します。これらの基準と手法は、総称してデータ識別子と呼ばれ、多くの国や地域の機密データタイプの大規模かつ増加しているリストを検出できます。Amazon SNS マネージドデータ識別子には、財務データ、個人健康情報 (PHI)、個人を特定できる情報 (PII) を保護するために、事前設定されたデータタイプが用意されています。また、カスタムデータ識別子を使用して、特定のユースケースに合わせた独自のデータ識別子を作成することもできます。

トピック

- Amazon SNS でのマネージドデータ識別子の使用
- Amazon SNS でのカスタムデータ識別子の使用

Amazon SNS でのマネージドデータ識別子の使用

トピック

- マネージドデータ識別子とは
- 機密データタイプ: 認証情報
- 機密データタイプ: デバイス
- 機密データタイプ: 財務情報
- 機密データタイプ: 個人の健康情報 (PHI)
- 機密データタイプ: 個人を特定できる情報 (PII)

マネージドデータ識別子とは

Amazon SNS マネージドデータ識別子は、特定の種類の機密データ (たとえば、クレジットカード番号、AWS シークレットアクセスキー、または特定の国またはリージョンのパスポート番号) を検出するように設計されています。データ保護ポリシーを作成すると、これらの識別子を使用してトピックを通過するメッセージを分析し、検出された場合にアクションを実行するように Amazon SNS を設定できます。

Amazon SNS は、マネージドデータ識別子を使用して、次のカテゴリの機密データを検出できます。

- プライベートキーや AWS シークレットアクセスキーなどの認証情報
- IP アドレスや MAC アドレスなどのデバイス識別子。
- クレジットカード番号などの財務情報
- 健康保険または医療識別番号などの PHI に関する医療情報。
- 個人情報 (運転免許証や社会保障番号など)

各カテゴリ内で、Amazon SNS は複数のタイプの機密データを検出できます。このセクションのトピックでは、各タイプとその検出に関連する要件をリスト化して説明します。各タイプでは、データを検出するように設計されたマネージドデータ識別子の一意の識別子 (ID) も示します。データ保護ポリシーを作成するとき、この ID を使用して、検出するメッセージデータ保護のマネージドデータ識別子を含めることができます。

キーワード要件

特定のタイプの機密データを検出するため、Amazon SNS ではデータの近くにあるキーワードをスキャンします。特定のタイプのデータに当てはまる場合、このセクションのその後のトピックでは、そのデータの特定のキーワード要件を示します。

キーワードでは大文字と小文字が区別されません。さらに、キーワードにスペースが含まれている場合、Amazon SNS は、スペースを含まないキーワードのバリエーションや、スペースではなくアンダースコア (_) またはハイフン (-) を含むキーワードのバリエーションを自動的に照合します。場合によっては、Amazon SNS はキーワードの一般的なバリエーションに対処するためにキーワードを拡張または短縮します。

機密データタイプの Amazon SNS マネージドデータ識別子

次のテーブルでは、Amazon SNSがマネージドデータ識別子を使用して検出できる認証情報、デバイス、財務、医療および個人健康情報 (PHI) のタイプの一覧とその説明を示しています。これらは、

個人を特定できる情報 (PII) としても認定される可能性のある特定のタイプのデータに加えたものです。

地域に依存するデータ識別子には、ダッシュ付きの識別子名と 2 文字のコード (ISO 3166-1 alpha-2) が必要です。例えば、 DriversLicense-US。

識別子	カテゴリ	国/言語
BankAccountNumber	金融	DE、ES、FR、GB、IT
CepCode	個人	BR
Cnpj	個人	BR
CpfCode	個人	BR
DriversLicense	個人	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	米国
ElectoralRollNumber	個人	GB
HealthInsuranceCardNumber	健康	EU
HealthInsuranceClaimNumber	健康	米国
HealthInsuranceNumber	健康	FR
HealthcareProcedureCode	健康	米国
IndividualTaxIdentificationNumber	個人	米国

識別子	カテゴリ	国/言語
InseeCode	個人	FR
MedicareBeneficiaryNumber	健康	米国
NationalDrugCode	健康	米国
NationalIdentificationNumber	個人	DE、ES、IT
NationalInsuranceNumber	個人	GB
NationalProviderId	健康	米国
NhsNumber	健康	GB
NieNumber	個人	ES
NifNumber	個人	ES
PassportNumber	個人	CA、DE、ES、 FR、GB、IT、US
PermanentResidenceNumber	個人	CA
PersonalHealthNumber	健康	CA
PhoneNumber	個人	BR、DE、ES、 FR、GB、IT、US
PostalCode	個人	CA
RgNumber	個人	BR
SocialInsuranceNumber	個人	CA
SSN	個人	es-US
TaxId	個人	DE、ES、FR、GB
ZipCode	個人	US

言語や地域に依存しないサポート対象の識別子

識別子	カテゴリ
Address	個人
AwsSecretKey	認証情報
CreditCardExpiration	金融
CreditCardNumber	金融
CreditCardSecurityCode	金融
EmailAddress	個人
IpAddress	個人
LatLong	個人
名前	個人
OpenSshPrivateKey	認証情報
PgpPrivateKey	認証情報
PkcsPrivateKey	認証情報
PuttyPrivateKey	認証情報
VehicleIdentificationNumber	個人

機密データタイプ: 認証情報

次のテーブルでは、Amazon SNS がマネージドデータ識別子を使用して検出できる認証情報のタイプをリスト化して説明します。

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	国とリージョン
AWS シークレットアクセスキー	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	すべて
OpenSSH プライベートキー	OpenSshPrivateKey	いいえ	すべて
PGP プライベートキー	PgpPrivateKey	いいえ	すべて
公開鍵暗号標準 (PKCS) プライベートキー	PkcsPrivateKey	いいえ	すべて
PuTTY プライベートキー	PuttyPrivateKey	いいえ	すべて

認証情報データタイプのデータ識別子 ARN

以下は、データ保護ポリシーに追加できるデータ識別子の Amazon リソースネーム (ARN) のリストを示しています。

認証情報データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/AwsSecretKey
arn:aws:dataprotection::aws:data-identifier/OpenSshPrivateKey
arn:aws:dataprotection::aws:data-identifier/PgpPrivateKey
arn:aws:dataprotection::aws:data-identifier/PkcsPrivateKey
arn:aws:dataprotection::aws:data-identifier/PuttyPrivateKey

機密データタイプ: デバイス

次のテーブルは、Amazon SNS がマネージドデータ識別子を使用して検出できるデバイス識別子のタイプをリスト化して説明します。

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	国とリージョン
IP アドレス	IpAddress	いいえ	すべて

デバイスデータタイプのデータ識別子 ARN

以下は、データ保護ポリシーに追加できるデータ識別子の Amazon リソースネーム (ARN) のリストを示しています。

デバイスデータ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/IpAddress

機密データタイプ: 財務情報

次のテーブルでは、Amazon SNS がマネージドデータ識別子を使用して検出できる財務情報のタイプをリスト化して説明します。

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
銀行口座番号	BankAccountNumber BankAccountNumber-US	はい、銀行口座番号のキーワード を参照してください。	これには、国コードなどの要素を含む、最大 34 文字の英数字で構成される国際銀行口座番号 (IBAN) が含まれます。	フランス、ドイツ、イタリア、スペイン、英国

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
クレジットカードの有効期限	CreditCardExpiration	exp d、exp m、exp y、expiration、expiry	–	すべて
クレジットカードの磁気ストライプデータ	CreditCardMagneticStripe	以下が含まれます: card data、iso7813、mag、magstripe、stripe、swipe。	トラック 1 と 2 が含まれます。	すべて

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
クレジットカード番号	CreditCardNumber	account number、american express、amex、bank card、card、card num、card number、cc #、ccn、check card、credit、credit card#、dankort、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	検出では、データは Luhn チェック式に準拠した 13～19 桁のシーケンスである必要があり、American Express、Dankort、Dinner'sTAK、Discover、truen、日本語カード局 (JCB)、TAK UnionPay、および Visa (1 以下のスーパースクリプトリンク) のいずれかの種類のクレジットカードに標準のカード番号プレフィックスを使用します。	すべて

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
クレジットカード認証コード	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	–	すべて

1. Amazon SNS は、クレジットカード発行会社がパブリックテストのために予約している、以下のシーケンスの出現をレポートしません。

   122000000000003、2222405343248877、2222990905257051、2223007648726984、22235771200176 および 76009244561。

銀行口座番号のキーワード

次のキーワードを使用して、国コードなどの要素を含む、最大 34 文字の英数字で構成される国際銀行口座番号 (IBAN) を検出します。

国またはリージョン	キーワード			
フランス	account code, account number, accountno #, accountnumber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte			
ドイツ	account code, account number, accountno #, accountnumber#, bankleitzahl, bban, customer account id, customer account number, customer bank account id, geheimzahl, iban, kartennummer, kontonummer, kreditkar			

国またはリージョン	キーワード			
	tennummer, sepa			
イタリア	account code, account number, accountno #, accountnumber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			

国またはリージョン	キーワード			
スペイン	account code, account number, accountno #, accountnumber#, 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			
英国	account code, account number, accountno #, accountnumber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			

国またはリージョン	キーワード			
米国	bank account、bank acct、checking account、checking acct、deposit account、deposit acct、savings account、savings acct、chequing account、chequing acct			

財務データタイプのデータ識別子 ARN

以下は、データ保護ポリシーに追加できるデータ識別子の Amazon リソースネーム (ARN) のリストを示しています。

財務データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-DE
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-ES
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-FR
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-GB
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-IT
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-US
arn:aws:dataprotection::aws:data-identifier/CreditCardExpiration
arn:aws:dataprotection::aws:data-identifier/CreditCardNumber

財務データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/CreditCardSecurityCode

機密データタイプ: 個人の健康情報 (PHI)

次のテーブルでは、Amazon SNS がマネージドデータ識別子を使用して検出できる保護対象の医療情報 (PHI) のタイプをリスト化して説明します。

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	国とリージョン
麻薬取締局 (DEA) 登録番号	DrugEnforcementAgencyNumber	dea number, dea registration	米国
健康保険証番号 (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 card number、health insurance card、health insurance number、insurance card number、krankenversicherungskarte、krankenversicherungsnummer、med	EU

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	国とリージョン
		ical 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 numero、versicherungsnummer	
健康保険請求番号 (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	US

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	国とリージョン
健康保険または医療識別番号	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR
ヘルスケア共通手順コーディングシステム (HCPCS) コード	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	US
メディケア受給者番号 (MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	US
全米医薬品コード (NDC)	NationalDrugCode	national drug code, ndc	US
国家プロバイダー識別子 (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	US
National Health Service (NHS) 番号	NhsNumber	national health service, NHS	GB
個人健康管理番号 (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

健康保険と医療識別番号のキーワード

さまざまなタイプの健康保険と医療識別番号を検出するために、Amazon SNS では番号の近くにあるキーワードが必要です。これには、欧州健康保険カード番号 (EU、フィンランド)、健康保険番号 (フランス)、メディケア受益者識別子 (米国)、国民保険番号 (英国)、NHS 番号 (英国)、個人保険番号 (カナダ) が含まれます。

次のテーブルに、Amazon SNS が特定の国およびリージョンについて認識するキーワードのリストを示します。

国またはリージョン	キーワード
カナダ	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
フィンランド	ehic, ehic#, finland health insurance card, finlandehicnumber#, finska sjukförsäkringskort, hälsokort, health card, health card number, health insurance card, health insurance number, sairaanhoitokortin, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskor

国またはリージョン	キーワード
	t, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti
フランス	carte d'assuré social, carte vitale, insurance card
英国	国民保健サービス、NHS
米国	mbi, medicare beneficiary

保護対象の医療情報 (PHI) データタイプのデータ識別子 ARN

PHI データ保護ポリシーで使用できるデータ識別子 Amazon リソースネーム (ARN) を以下に示します。

PHI データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/DrugEnforcementAgencyNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthcareProcedureCode-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceCardNumber-EU
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceClaimNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceNumber-FR
arn:aws:dataprotection::aws:data-identifier/MedicareBeneficiaryNumber-US
arn:aws:dataprotection::aws:data-identifier/NationalDrugCode-US
arn:aws:dataprotection::aws:data-identifier/NationalInsuranceNumber-GB
arn:aws:dataprotection::aws:data-identifier/NationalProviderId-US
arn:aws:dataprotection::aws:data-identifier/NhsNumber-GB
arn:aws:dataprotection::aws:data-identifier/PersonalHealthNumber-CA

機密データタイプ: 個人を特定できる情報 (PII)

次の表では、Amazon SNS がマネージドデータ識別子を使用して検出できる個人を特定できる情報 (PII) のタイプをリスト化して説明します。

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
生年月日	DateOfBirth	dob, date of birth, birthdate, birth date, birthday, b-day, bday	Support には、すべての数字や数字と月の名前の組み合わせなど、ほとんどの日付形式が含まれます。日付コンポーネントは、スペース、スラッシュ (/)、またはハイフン (-) で区切ることができます。	すべて
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	ブラジル
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	ブラジル

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	ブラジル

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
運転免許証識別番号	DriversLicense	はい、運転免許証識別番号のキーワード を参照してください。	–	オーストラリア、オーストリア、ベルギー、ブルガリア、カナダ、クロアチア、キプロス、チェコ共和国、デンマーク、エストニア、フィンランド、フランス、ドイツ、ギリシャ、ハンガリー、アイルランド、イタリア、ラトビア、リトアニア、ルクセンブルク、マルタ、オランダ、ポーランド、ポルトガル、ルーマニア、スロバキア、スロベニア、スペイン、スウェーデン、英国、米国

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
選挙人名簿番号	ElectoralRollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	UK
個人納税者識別	IndividualTaxIdentificationNumber	はい、納税者識別と参照番号のキーワード を参照してください。	–	米国
国立統計経済研究所 (INSEE)	InseeCode	はい、国民識別番号のキーワード を参照してください。	–	フランス
国民識別番号	NationalIdentificationNumber	はい、国民識別番号のキーワード を参照してください。	これには、Documento Nacional de Identidad (DNI) 識別子 (スペイン)、Codice fiscale codes (イタリア)、国民 ID カード番号 (ドイツ語) が含まれます。	ドイツ、イタリア、スペイン

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
国民保険番号 (NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationalinsurance#, nationalinsurancenumber, nin, nino	–	UK
Número de identidad de extranjero (NIE)	NieNumber	はい、納税者識別と参照番号のキーワード を参照してください。	–	スペイン
Número de Identificación Fiscal (NIF)	NifNumber	はい、納税者識別と参照番号のキーワード を参照してください。	–	スペイン
パスポート番号	PassportNumber	はい、パスポート番号のキーワード を参照してください。	–	カナダ、フランス、ドイツ、イタリア、スペイン、英国、米国

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
本籍地	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	–	カナダ

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
電話番号	PhoneNumber	ブラジル: キーワードには cel、celular、fone、móvel、número residencial、numero residencial、telefone も含まれます その他: cell、contact、fax、fax number、mobile、phone、phone number、tel、telephone、telephone number	これには、米国の通話料無料の番号とファックス番号が含まれます。キーワードがデータの近くにある場合、番号に国コードを含める必要はありません。キーワードがデータの近くにない場合は、番号に国コードを含める必要があります。	ブラジル、カナダ、フランス、ドイツ、イタリア、スペイン、英国、米国
郵便番号	PostalCode	No	–	カナダ
Registro Geral (RG)	RgNumber	はい、国民識別番号のキーワード を参照してください。	–	ブラジル

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
社会保険番号 (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	カナダ
社会保障番号 (SSN)	Ssn	スペイン – número de la seguridad social、social security no.、social security no. número de la seguridad social、social security number、socialsecurityno#、ssn、ssn# US — social social、ss#、ssn	–	スペイン、米国
納税者識別番号または参照番号	TaxId	はい、納税者識別と参照番号のキーワード を参照してください。	これには、TIN (フランス)、Steueridentifikationsnummer (ドイツ)、CIF (スペイン)、TRN、UTR (英国) が含まれます。	フランス、ドイツ、スペイン、英国

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
米国郵便番号	ZipCode	zip code, zip+4	–	US
郵送先住所	Address	No	キーワードは必要ありませんが、検出では、住所には都市または場所の名前および郵便番号を含める必要があります。	オーストラリア、カナダ、フランス、ドイツ、イタリア、スペイン、英国、米国
電子メールアドレス	EmailAddress	email、email address、e mail、e mail address	–	すべて

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
全地球測位システム (GPS) 座標	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS は、緯度と経度の座標がペアとして保存され、それらが10 進度 (DD) 形式の場合、GPS 座標を検出できます (例: 41.948614 ,-87.655311)。サポートには、度 10 進分 (DDM) 形式 (例: 41°56.9168'N 87°39.3187'W)、または、度、分、秒 (DMS) 形式 (例: 41°56'55.0104"N 87°39'19.1196"W) の座標は含まれません。	すべて
フルネーム	Name	No	Amazon SNS はフルネームのみを検出できます。Support はラテン文字セットに限定されます。	すべて

検出タイプ	マネージドデータ識別子 ID	キーワードが必須	追加情報	国とリージョン
車両識別番号 (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	Amazon SNS は、17 文字のシーケンスで構成され、ISO 3779 および 3780 規格に従った VIN を検出できます。これらの規格は、世界中で使用するために設計されています。	すべて

運転免許証識別番号のキーワード

さまざまなタイプの運転免許証識別番号を検出するために、Amazon SNS では番号の近くにあるキーワードが必要です。次のテーブルに、Amazon SNS が特定の国およびリージョンについて認識するキーワードのリストを示します。

国またはリージョン	キーワード
オーストラリア	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

国またはリージョン	キーワード
オーストリア	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
ベルギー	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
ブルガリア	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
カナダ	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
クロアチア	vozačka dozvola
キプロス	άδεια οδήγησης
チェコ共和国	číslo licence, číslo licence řidiče, číslo řidičského průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
デンマーク	kørekort, kørekortnummer

国またはリージョン	キーワード
エストニア	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
フィンランド	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
フランス	permis de conduire
ドイツ	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
ギリシャ	δεια οδήγησης, adeia odigisis
ハンガリー	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
アイルランド	ceadúnas tiomána
イタリア	patente di guida, patente di guida numero, patente guida, patente guida numero
ラトビア	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
リトアニア	vairuotojo pažymėjimas
ルクセンブルグ	fahrerlaubnis, führerschäin
マルタ	liċenzja tas-sewqan
オランダ	permis de conduire, rijbewijs, rijbewijsnummer

国またはリージョン	キーワード
ポーランド	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
ポルトガル	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
ルーマニア	numărul permisului de conducere, permis de conducere
スロバキア	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
スロベニア	vozniško dovoljenje
スペイン	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
スウェーデン	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

国またはリージョン	キーワード
英国	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
米国	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

国民識別番号のキーワード

さまざまなタイプの国民識別番号を検出するために、Amazon SNS では番号の近くにあるキーワードが必要です。これには、Documento Nacional de Identidad (DNI) 識別子 (スペイン)、フランス国立統計経済研究所 (INSEE) コード、ドイツの国民 ID カード番号、Registro Geral (RG) 番号 (ブラジル) が含まれます。

次のテーブルに、Amazon SNS が特定の国およびリージョンについて認識するキーワードのリストを示します。

国またはリージョン	キーワード
ブラジル	registro geral, rg
フランス	assurance sociale, carte nationale d’identité, cni, code sécurité sociale, French social security number, fssn#, insee, insurance

国またはリージョン	キーワード
	number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
ドイツ	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
イタリア	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
スペイン	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#

パスポート番号のキーワード

さまざまなタイプのパスポート番号を検出するには、Amazon SNS では番号の近くにあるキーワードが必要です。次のテーブルに、Amazon SNS が特定の国およびリージョンについて認識するキーワードのリストを示します。

国またはリージョン	キーワード
カナダ	passeport, passeport#, passport, passport#, passportno, passportno#
フランス	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non

国またはリージョン	キーワード
ドイツ	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
イタリア	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
スペイン	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
英国	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
米国	passport, travel document

納税者識別と参照番号のキーワード

さまざまなタイプの納税者識別番号と参照番号を検出するために、Amazon SNS では番号の近くにあるキーワードが必要です。次のテーブルに、Amazon SNS が特定の国およびリージョンについて認識するキーワードのリストを示します。

国またはリージョン	キーワード
ブラジル	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

国またはリージョン	キーワード
フランス	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
ドイツ	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
スペイン	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#
英国	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
米国	個別の納税者識別番号、itin、i.t.i.n。

個人を特定できる情報 (PII) のデータ識別子 ARN

以下の表は、データ保護ポリシーに追加できるデータ識別子の Amazon リソースネーム (ARN) のリストを示しています。

PII データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:dataprotection::aws:data-identifier/CepCode-BR
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn:aws:dataprotection::aws:data-identifier/CpfCode-BR

PII データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/DateOfBirth
arn:aws:dataprotection::aws:data-identifier/DriversLicense-AT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-AU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-BE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-BG
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CA
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CY
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CZ
arn:aws:dataprotection::aws:data-identifier/DriversLicense-DE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-DK
arn:aws:dataprotection::aws:data-identifier/DriversLicense-EE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-ES
arn:aws:dataprotection::aws:data-identifier/DriversLicense-FI
arn:aws:dataprotection::aws:data-identifier/DriversLicense-FR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-GB
arn:aws:dataprotection::aws:data-identifier/DriversLicense-GR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-HR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-HU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-IE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-IT

PII データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LV
arn:aws:dataprotection::aws:data-identifier/DriversLicense-MT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-NL
arn:aws:dataprotection::aws:data-identifier/DriversLicense-PL
arn:aws:dataprotection::aws:data-identifier/DriversLicense-PT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-RO
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SI
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SK
arn:aws:dataprotection::aws:data-identifier/DriversLicense-US
arn:aws:dataprotection::aws:data-identifier/ElectoralRollNumber-GB
arn:aws:dataprotection::aws:data-identifier/EmailAddress
arn:aws:dataprotection::aws:data-identifier/IndividualTaxIdentificationNumber-US
arn:aws:dataprotection::aws:data-identifier/InseeCode-FR
arn:aws:dataprotection::aws:data-identifier/LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-DE
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-ES

PII データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-IT
arn:aws:dataprotection::aws:data-identifier/NieNumber-ES
arn:aws:dataprotection::aws:data-identifier/NifNumber-ES
arn:aws:dataprotection::aws:data-identifier/PassportNumber-CA
arn:aws:dataprotection::aws:data-identifier/PassportNumber-DE
arn:aws:dataprotection::aws:data-identifier/PassportNumber-ES
arn:aws:dataprotection::aws:data-identifier/PassportNumber-FR
arn:aws:dataprotection::aws:data-identifier/PassportNumber-GB
arn:aws:dataprotection::aws:data-identifier/PassportNumber-IT
arn:aws:dataprotection::aws:data-identifier/PassportNumber-US
arn:aws:dataprotection::aws:data-identifier/PermanentResidenceNumber-CA
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-BR
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-DE
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-ES
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-FR
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-GB
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-IT
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US
arn:aws:dataprotection::aws:data-identifier/PostalCode-CA
arn:aws:dataprotection::aws:data-identifier/RgNumber-BR

PII データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/SocialInsuranceNumber-CA
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:aws:dataprotection::aws:data-identifier/TaxId-DE
arn:aws:dataprotection::aws:data-identifier/TaxId-ES
arn:aws:dataprotection::aws:data-identifier/TaxId-FR
arn:aws:dataprotection::aws:data-identifier/TaxId-GB
arn:aws:dataprotection::aws:data-identifier/VehicleIdentificationNumber
arn:aws:dataprotection::aws:data-identifier/ZipCode-US

Amazon SNS でのカスタムデータ識別子の使用

トピック

SNS カスタムデータ識別子とは

カスタムデータ識別子 (CDI) を使用すると、データ保護ポリシーで使用できる独自のカスタム正規表現を定義できます。カスタムデータ識別子を使用すると、マネージドデータ識別子では提供できないビジネス固有の個人を特定できる情報 (PII) のユースケースをターゲットにすることができます。たとえば、カスタムデータ識別子を使用すると、会社固有の従業員 ID を検索できます。カスタムデータ識別子は、マネージドデータ ID と組み合わせて使用できます。

データ保護ポリシーでカスタムデータ識別子を使用する

以下のデータ保護ポリシーでは、会社固有の従業員 ID を含むペイロードを検出し、これらの ID をハッシュ記号 (#) でマスクするよう Amazon SNS トピックに指示します。

1. データ保護ポリシー内で Configuration ブロックを作成します。
2. カスタムデータ識別子の Name を入力します。例えば、「**EmployeeId**」と入力します。
3. カスタムデータ識別子の Regex を入力します。例えば、「**EID-\d{9}-US**」と入力します。
4. ポリシーステートメントで、以下のカスタムデータ識別子を参照します。

```
{
 "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. (オプション) 必要に応じて、Configuration ブロックにさらに カスタムデータ識別子を追加します。現在、データ保護ポリシーでは最大 10 個のカスタムデータ識別子を使用できます。

カスタムデータ識別子の制約

Amazon SNS カスタムデータ識別子には、以下の制限があります。

- 各データ保護ポリシーに使用できるカスタムデータ識別子は最大 10個です。
- カスタムデータ識別子名に使用できるのは 128 文字までです。以下の文字がサポートされています。
  - 英数字: (a-zA-Z0-9)
  - 記号: ('_' | '-')
- RegEx の最大長は 200 文字です。以下の文字がサポートされています。
  - 英数字: (a-zA-Z0-9)
  - 記号: ('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | ' ')
  - RegEx 予約文字: ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' | '\\' | '*' | '+' | '.')
- カスタムデータ識別子は、マネージドデータ識別子と同じ名前を共有することはできません。
- カスタムデータ識別子は、各 Amazon SNS トピックのすべてのデータ保護ポリシーで指定する必要があります。

Amazon SNS メッセージ配信

このセクションでは、メッセージ配信の仕組みについて説明します。

トピック

Amazon SNS raw メッセージの配信

Amazon Data Firehose 、Amazon SQS、および HTTP/S エンドポイントがメッセージの JSON フォーマットを処理するのを避けるため、Amazon SNS は raw メッセージ配信を許可します。

- Amazon Data Firehose または Amazon SQS エンドポイントの raw メッセージ配信を有効にすると、発行されたメッセージから Amazon SNS メタデータが削除され、メッセージはそのまま送信されます。
- HTTP/S エンドポイントに対して raw メッセージの配信を有効にすると、値が `true` に設定された HTTP ヘッダー `x-amz-sns-rawdelivery` がメッセージに追加されます。これは、メッセージが JSON フォーマットなしで発行されたことを示します。
- HTTP/S エンドポイントに対して raw メッセージの配信を有効にすると、メッセージ本文、クライアント IP、および必要なヘッダーが配信されます。メッセージの属性を指定すると、そのメッセージは送信されません。
- Firehose エンドポイントの raw メッセージ配信を有効にすると、メッセージ本文が配信されます。メッセージの属性を指定すると、そのメッセージは送信されません。

AWS SDK を使用して raw メッセージの配信を有効にするには、 `SetSubscriptionAttribute` API アクションを使用し、 `RawMessageDelivery` 属性の値を に設定する必要がありますtrue。

AWS Management Consoleを使用して raw メッセージ配信を有効にする

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. トピックページで、Firehose、Amazon SQS、または HTTP/S エンドポイントにサブスクライブしているトピックを選択します。
4. ***MyTopic*** ページのサブスクリプションセクションで、サブスクリプションを選択し、 の編集を選択します。
5. [Edit ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***] ページの [詳細] セクションで、[raw メッセージ配信の有効化] を選択します。
6. [変更を保存] をクリックします。

メッセージ形式の例

以下は、同じメッセージが同じ Amazon SQS キューに 2 回送信されている例です。唯一の違いは、最初のメッセージでは raw メッセージ配信が無効になり、2 番目のメッセージでは有効になっていることです。

- raw メッセージの配信を無効にします。


```
{
  "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"
}
```

- raw メッセージの配信を有効にします。

```
This is a test message.
```

Amazon SQS サブスクリプションのメッセージ属性と raw メッセージ配信

Amazon SNS では、メッセージ属性の配信がサポートされています。これにより、タイムスタンプ、地理空間データ、署名、識別子などの構造化メタデータ項目をメッセージについて提供できます。Raw Message Delivery が有効になっている Amazon SQS サブスクリプションの場合、最大 10 個のメッセージ属性を送信できます。10 個を超えるメッセージ属性を送信するには、Raw メッセージの配信を無効にする必要があります。ただし、Amazon SNS は、raw メッセージ配信が有効になっている Amazon SQS サブスクリプションに向けられた 10 を超えるメッセージ属性を持つメッセージを破棄し、クライアント側のエラーとして扱います。

別のアカウントの Amazon SQS キューへ Amazon SNS メッセージを送信する

このドキュメントでは、別のアカウントの Amazon SQS キューに 1 つ以上のサブスクリプションがある Amazon SNS トピックに通知を発行する方法について説明します。同じアカウント内にある場合と同じ方法でトピックとキューをセットアップします (「Amazon SQS キューへのファンアウト」を参照)。主な相違点はサブスクリプションの確認の処理方法で、トピックへのキューのサブスクライブ方法によって異なります。

キューの所有者がサブスクリプションを作成すると確認が自動的に実行されるため、可能な場合は「キューの所有者がサブスクリプションを作成する」セクションで参照されている手順に従うことがベストプラクティスです。

> **Note**
>
> Amazon SQS キューに大量のメッセージがある場合は、キュー所有者がサブスクリプションを作成することをお勧めします。

トピック

- キューの所有者がサブスクリプションを作成する
- キュー作成サブスクリプションを所有していないユーザー
- サブスクリプション解除リクエストでサブスクリプションに認証を要求させるにはどうすればよいですか。

キューの所有者がサブスクリプションを作成する

Amazon SQS キューを作成したアカウントがキューの所有者です。キューの所有者がサブスクリプションを作成するとき、サブスクリプションを確認する必要はありません。Subscribe アクションが完了するとすぐに、キューはトピックからの通知の受信を開始します。キューの所有者がトピック所有者のトピックにサブスクライブできるようにするには、トピック所有者が、トピックに対して Subscribe アクションを呼び出す許可をキューの所有者のアカウントに付与する必要があります。

ステップ 1: AWS Management Console を使用してトピックポリシーを設定するには

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. トピックを選択し、[編集] を選択します。
4. [Edit ***MyTopic***] ページで [アクセスポリシー] セクションを展開します。
5. 以下のポリシーを入力します。

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "111122223333"
         },
         "Action": "sns:Subscribe",
         "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
   ]
}
```

このポリシーは、123456789012 アカウント内で MyTopic に sns:Subscribe を呼び出す許可を 111122223333 アカウントに付与します。

111122223333 のアカウントに対する認証情報が持っているユーザーは、MyTopic に登録できます。この許可によって、アカウント ID は、アクセス許可 を IAM ユーザー/ロールに委任できます。呼び出しは、ルートアカウントまたは管理者ユーザーのみに許可されます。また、IAM ユーザー/ロールは、キューにサブスクリプションを許可するために sns:subscribe を持つ必要があります。

6. [Save changes] (変更の保存) をクリックします。

   111122223333 アカウントに対する認証情報があるユーザーは、MyTopic に登録できます。

ステップ 2: AWS Management Console を使用して別の AWS アカウント のトピックに Amazon SQS キューのサブスクリプションを追加するには

開始する前に、トピックおよびキューの ARN を保持していて、キューにメッセージを送信する許可がトピックに付与されていることを確認してください。

1. Amazon SQSコンソールにサインインします。
2. ナビゲーションパネルで、[Queues](キュー) を選択します。
3. キューのリストからキューを選択して、Amazon SNS トピックを購読します。
4. [Subscribe to Amazon SNS topic] (Amazon SNS トピックを購読する) を選択します。
5. [Specify an Amazon SNS topic available for this queue menu] (このキューメニューで使用できる Amazon SNS トピックを指定する) で、キューの Amazon SNS topic (Amazon SNS トピック) を選択します。
6. [Amazon SNS トピックの ARN を入力する]を選択してから、トピックの Amazon リソースネーム (ARN) を入力します。
7. [Save (保存)] を選択します。

> **Note**
>
> - サービスと通信できるようにするには、キューに Amazon SNS へのアクセス権限が必要です。
> - キューの所有者であるため、サブスクリプションを確認する必要はありません。

キュー作成サブスクリプションを所有していないユーザー

サブスクリプションを作成するユーザーがキューの所有者ではない場合、そのサブスクリプションを確認する必要があります。

`Subscribe` アクションを使用すると、Amazon SNS はサブスクリプションの確認をキューに送信します。サブスクリプションが Amazon SNS コンソールに表示され、サブスクリプション ID が [保留中の確認] に設定されます。

サブスクリプションを確認するには、キューからメッセージを読み取る権限を持つユーザーがサブスクリプションの確認 URL を取得し、サブスクリプションの所有者がサブスクリプションの確認 URL を使用してサブスクリプションを確認する必要があります。サブスクリプションが確認されるまで、トピックに対して発行された通知はキューに送信されません。サブスクリプションを確認するには、Amazon SQS コンソールまたは `ReceiveMessage` アクションを使用できます。

> **Note**
>
> トピックにエンドポイントをサブスクライブする前に、キューに `sqs:SendMessage` アクセス権限を設定してこのキューがトピックからメッセージを受信できるようにする必要があります。詳細については、「ステップ 2: Amazon SQS キューにメッセージを送信する許可を Amazon SNS トピックに付与する」を参照してください。

ステップ 1: AWS Management Console を使用して別の AWS アカウント のトピックに Amazon SQS キューのサブスクリプションを追加するには

開始する前に、トピックおよびキューの ARN を保持していて、キューにメッセージを送信する許可がトピックに付与されていることを確認してください。

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[サブスクリプション] を選択します。
3. [サブスクリプション] ページで [サブスクリプションの作成] を選択します。
4. [サブスクリプションの作成] ページで [詳細] セクションで、以下を実行します。
   a. [トピック ARN] にトピックの ARN を入力します。
   b. [プロトコル] で、[Amazon SQS] を選択します。
   c. [エンドポイント] にキューの ARN を入力します。

d. [サブスクリプションの作成] を選択します。

> **Note**
>
> - サービスと通信できるようにするには、キューに Amazon SNS へのアクセス権限が必要です。

Amazon SNS トピックが Amazon SQS キューへメッセージを送信できるようにするポリシーステートメントの例を以下に示します。

```
{
   "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"
      }
   }
}
```

ステップ 2: AWS Management Console コンソールを使用してサブスクリプションを確認するには

1. Amazon SQS コンソールにサインインします。
2. トピックへのサブスクリプションが保留になっているキューを選択します。
3. [Send and receive messages] (メッセージの送受信) を選択し、[Poll for messages] (メッセージのポーリング) を選択します。

   サブスクリプションの確認を含むメッセージがキューで受信されます。
4. [本文] 列で、次の操作を行います。

   a. [詳細] を選択します。

   b. [Message Details] (メッセージの詳細) ダイアログボックスで、[SubscribeURL] の値を見つけてメモします。これはサブスクリプションリンクです (以下に例を示し

ます)。API トークンの検証の詳細については、Amazon SNS API リファレンスの「ConfirmSubscription」を参照してください。

```
https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
east-2:123456789012:MyTopic&Token=2336412f37fb...
```

c. サブスクリプション確認リンクを書き留めます。URL は、キュー所有者からサブスクリプション所有者に渡す必要があります。サブスクリプションの所有者は、URL を Amazon SNS コンソールに入力する必要があります。

5. サブスクリプション所有者として Amazon SNS コンソールにログインします。サブスクリプション所有者が確認を実行します。
6. 関連する [topic] (トピック) を選択します。
7. トピックのサブスクリプション一覧表で、関連するサブスクリプションを選択します。「Pending confirmation (保留中の確認)」とラベル付けされています。
8. [confirm subscription] (サブスクリプションの確認) を選択します。
9. サブスクリプション確認リンクを求めるモーダルが表示されます。サブスクリプション確認リンクを貼り付けます。
10. モーダルで [Confirm subscription] (サブスクリプションの確認) を選択します。

    XML 応答が表示されます。例えば、

```
<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>
```

    サブスクライブしたキューは、トピックからメッセージを受信する準備ができています。
11. (オプション) Amazon SNS コンソールでトピックのサブスクリプションを表示すると、[保留中の確認] メッセージが [サブスクリプション ID] 列でサブスクリプション ARN に置き換えられています。

サブスクリプション解除リクエストでサブスクリプションに認証を要求させるにはどうすればよいですか。

サブスクリプションの所有者は、サブスクリプション確認時に AuthenticateOnUnsubscribe フラグを true に設定する必要があります。

- キューの所有者がサブスクリプションを作成すると、AuthenticateOnUnsubscribe は自動的に true に設定されます。
- 認証なしでサブスクリプション確認リンクに移動した場合、AuthenticateOnUnsubscribe を true に設定することはできません。

Amazon SQS キューまたは別のリージョンでの AWS Lambda 関数へ Amazon SNS メッセージを送信する

Amazon SNS はクロスリージョン配信をサポートしています。これは、デフォルトで有効になっているリージョンとオプトインリージョンです。Amazon SNS がサポートする AWS リージョン (オプトインリージョンを含む) の最新のリストについては、「Amazon Web Services 全般のリファレンス」の「Amazon Simple Notification Service エンドポイントとクォータ」を参照してください。

Amazon SNS は、Amazon SQS キューへの通知のクロスリージョン配信と AWS Lambda 関数をサポートしています。リージョンの 1 つがオプトインリージョンである場合は、サブスクライブされたリソースのポリシーで別の Amazon SNS サービスプリンシパルを指定する必要があります。

Amazon SNS サブスクリプションコマンドは、Amazon SNS がホストされているリージョンのターゲットアカウントで実行する必要があります。例えば、Amazon SNS が us-east-1 リージョンのアカウント「A」にあり、Lambda 関数が us-east-2 リージョンのアカウント「B」にある場合、サブスクリプション CLI コマンドは us-east-1 リージョンのアカウント「A」で実行する必要があります。

オプトインリージョン

Amazon SNS では、次のオプトインリージョンがサポートされています。

リージョン名	リージョン
アフリカ (ケープタウン) リージョン	af-south-1
アジアパシフィック (香港) リージョン	ap-east-1

リージョン名	リージョン
アジアパシフィック (ハイデラバード) リージョン	ap-south-2
アジアパシフィック (ジャカルタ) リージョン	ap-southeast-3
アジアパシフィック (メルボルン) リージョン	ap-southeast-4
アジアパシフィック (大阪) リージョン	ap-northeast-3
欧州 (ミラノ) リージョン	eu-south-1
欧州 (スペイン) リージョン	eu-south-2
欧州 (チューリッヒ) リージョン	eu-central-2
イスラエル (テルアビブ) リージョン	il-central-1
中東 (バーレーン) リージョン	me-south-1
中東 (アラブ首長国連邦) リージョン	me-central-1

オプトインリージョンを有効にする方法については、「」の「 AWS リージョンの管理」を参照してくださいAmazon Web Services 全般のリファレンス。

Amazon SNS を使用してオプトインリージョンからデフォルトで有効になっているリージョンにメッセージを配信する場合、キュー用に作成されたリソースポリシーを変更する必要があります。プリンシパル `sns.amazonaws.com` を `sns.<opt-in-region>.amazonaws.com` に置き換えます。例:

- 例えば、米国東部 (バージニア北部) の Amazon SQS キューをアジアパシフィック (香港) の Amazon SNS トピックにサブスクライブする場合は、キューポリシーのプリンシパルを `sns.ap-east-1.amazonaws.com` に変更します。オプトインリージョンには、2019 年 3 月 20 日以降に開始されたリージョンが含まれます。これには、アジアパシフィック (香港)、アジアパシフィック (ジャカルタ)、中東 (バーレーン)、欧州 (ミラノ)、アフリカ (ケープタウン) が含まれます。2019 年 3 月 20 日以前に開始されたリージョンは、デフォルトで有効になっています。

Amazon SQS へのリージョン間での配信サポート

リージョン間での配信	サポート対象/サポート対象外	
デフォルトで有効なリージョンからオプトインリージョン	キューのサービスプリンシパルで sns.<opt-in-region>.amazonaws.com を使用してサポート	
オプトインリージョンからデフォルトで有効なリージョン	キューのサービスプリンシパルで sns.<opt-in-region>.amazonaws.com を使用してサポート	
オプトインリージョンからオプトインリージョン	サポートされていません	

以下は、オプトインリージョン (af-south-1) の Amazon SNS トピックが enabled-by-default リージョン (us-east-1) の Amazon SQS キューに配信できるようにするアクセスポリシーステートメントの例です。これには、パス (Statement/Principal/Service) の下に、必要なリージョン化されたサービスプリンシパル設定が含まれています。

```
{
  "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"
        }
```

```
      }
    },
    ...
  ]
}
```

- 米国東部 (バージニア北部) の AWS Lambda 関数をアジアパシフィック (香港) の Amazon SNS トピックにサブスクライブするには、 AWS Lambda 関数ポリシーのプリンシパルを に変更しますsns.ap-east-1.amazonaws.com。オプトインリージョンには、2019 年 3 月 20 日以降に開始されたリージョンが含まれます。これには、アジアパシフィック (香港)、アジアパシフィック (ジャカルタ)、中東 (バーレーン)、欧州 (ミラノ)、アフリカ (ケープタウン) が含まれます。2019 年 3 月 20 日以前に開始されたリージョンは、デフォルトで有効になっています。

  へのクロスリージョン配信のサポート AWS Lambda

リージョン間での配信	サポート対象/サポート対象外	
デフォルトで有効なリージョンからオプトインリージョン	サポートされていません	
オプトインリージョンからデフォルトで有効なリージョン	Lambda 関数のサービスプリンシパルで sns.<opt-in-region>.amazonaws.com を使用してサポート	
オプトインリージョンからオプトインリージョン	サポートされていません	

Amazon SNS メッセージ配信ステータス

Amazon SNS では、以下の Amazon SNS エンドポイントでトピックに送信された通知メッセージの配信ステータスをログに記録できます。

- HTTP
- Amazon Data Firehose
- AWS Lambda

- プラットフォームアプリケーションエンドポイント
- Amazon Simple Queue Service

メッセージ配信ステータス属性を設定すると、トピックサブスクライバーに送信されるメッセージのログエントリが CloudWatch Logs に送信されます。メッセージの配信ステータスを記録すると、以下のようにオペレーションをより把握することに役立ちます。

- メッセージが Amazon SNS エンドポイントに配信されたかどうかを知ることができます。
- Amazon SNS エンドポイントから Amazon SNS に送信された応答を識別します。
- メッセージのドウェル時間 (発行のタイムスタンプから Amazon SNS エンドポイントへの配信直前までの時間) を決定します。

メッセージの配信ステータスのトピック属性を設定するには、、 AWS Software Development Kit (SDKs) AWS Management Console、クエリ API、または を使用できます AWS CloudFormation。

トピック

- AWS Management Consoleを使用した配信ステータスのログ記録を設定する
- AWS SDKs を使用した配信ステータスのログ記録の設定
- AWS トピック属性を設定する SDK の例
- AWS CloudFormationを使用した配信ステータスのログ記録を設定する

AWS Management Consoleを使用した配信ステータスのログ記録を設定する

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. [トピック] ページで、トピックを選択して [編集] を選択します。
4. 編集***MyTopic***ページで、配信ステータスのログ記録セクションを展開します。
5. 配信ステータスをログに記録するプロトコルを選択します (AWS Lambda など)。
6. 成功のサンプルレート (CloudWatch ログを受信する成功メッセージの割合) を入力します。
7. [IAM ロール] セクションで、次のいずれかの操作を行います。
    - アカウントから既存のサービスロールを選択するには、[既存のサービスロールを使用] を選択して成功および失敗した配信の IAM ロールを指定します。

- アカウントに新しいサービスを作成するには、[新しいサービスロールを作成]、[新しいロールの作成] の順に選択して、IAM コンソールで成功および失敗した配信の IAM ロールを定義します。

  ユーザーに代わって CloudWatch ログを使用するための書き込みアクセスを Amazon SNS に付与するには、 を許可 を選択します。

8. [変更を保存] を選択します。

これで、メッセージの配信ステータスを含む CloudWatch ログを表示して解析できます。の使用の詳細については CloudWatch、「 CloudWatchドキュメント」を参照してください。

AWS SDKs を使用した配信ステータスのログ記録の設定

AWS SDKs、Amazon SNS でメッセージ配信ステータス属性を使用するための APIs を複数の言語で提供します。

トピック属性

メッセージの配信ステータスには、以下のトピック属性名の値を使用できます。

HTTP

- `HTTPSuccessFeedbackRoleArn` — HTTP エンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が成功したことを示します。
- `HTTPSuccessFeedbackSampleRate` — HTTP エンドポイントにサブスクライブされている Amazon SNS トピックについてサンプリングする、成功したメッセージの割合を示します。
- `HTTPFailureFeedbackRoleArn` — HTTP エンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が失敗したことを示します。

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn` — Amazon Kinesis Data Firehose エンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が成功したことを示します。
- `FirehoseSuccessFeedbackSampleRate` — Amazon Kinesis Data Firehose エンドポイントにサブスクライブされている Amazon SNS トピックについてサンプリングする、成功したメッセージの割合を示します。

- `FirehoseFailureFeedbackRoleArn` — Amazon Kinesis Data Firehose エンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が失敗したことを示します。

AWS Lambda

- `LambdaSuccessFeedbackRoleArn` — Lambda エンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が成功したことを示します。
- `LambdaSuccessFeedbackSampleRate` — Lambda エンドポイントにサブスクライブされている Amazon SNS トピックについてサンプリングする、成功したメッセージの割合を示します。
- `LambdaFailureFeedbackRoleArn` — Lambda エンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が失敗したことを示します。

プラットフォームアプリケーションエンドポイント

- `ApplicationSuccessFeedbackRoleArn` — AWS アプリケーションエンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が成功したことを示します。
- `ApplicationSuccessFeedbackSampleRate` — AWS アプリケーションエンドポイントにサブスクライブされている Amazon SNS トピックについてサンプリングする、成功したメッセージの割合を示します。
- `ApplicationFailureFeedbackRoleArn` — AWS アプリケーションエンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が失敗したことを示します。

> **Note**
>
> Amazon SNS アプリケーションエンドポイントに送信される通知メッセージのメッセージ配信ステータスのトピックの属性を設定できることに加えて、プッシュ通知サービスに送信されるプッシュ通知メッセージの配信ステータスのアプリケーション属性を設定することもできます。詳細については、「メッセージの配信ステータスの Amazon SNS アプリケーション属性を使用する」を参照してください。

Amazon SQS

- `SQSSuccessFeedbackRoleArn` — Amazon SQS エンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が成功したことを示します。

- SQSSuccessFeedbackSampleRate — Amazon SQS エンドポイントにサブスクライブされている Amazon SNS トピックについてサンプリングする、成功したメッセージの割合を示します。
- SQSFailureFeedbackRoleArn — Amazon SQS エンドポイントにサブスクライブされている Amazon SNS トピックのメッセージ配信が失敗したことを示します。

> **Note**
>
> <ENDPOINT>SuccessFeedbackRoleArn および <ENDPOINT>FailureFeedbackRoleArn 属性は、ユーザーに代わって CloudWatch Logs を使用するための書き込みアクセスを Amazon SNS に付与するために使用されます。<ENDPOINT>SuccessFeedbackSampleRate 属性は、正常な配信メッセージのサンプルレートの割合 (0〜100) を指定するためのものです。<ENDPOINT>FailureFeedbackRoleArn 属性を設定すると、失敗したメッセージ配信はすべて CloudWatch ログを生成します。

AWS トピック属性を設定する SDK の例

以下のコード例は、SetTopicAttributes の使用方法を示しています。

CLI

AWS CLI

トピックの属性を設定するには

次の set-topic-attributes の例では、指定したトピックの DisplayName 属性を設定します。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

このコマンドでは何も出力されません。

- API の詳細については、「 コマンドリファレンスSetTopicAttributes」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> の詳細については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスSetTopicAttributes」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> の詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスSetTopicAttributes」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> の詳細については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、SetTopicAttributesAWS 「 SDK for Kotlin API リファレンス」の「」を参照してください。

PHP

SDK for PHP

> **Note**
>
> の詳細については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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());
}
```

- API の詳細については、「 API リファレンスSetTopicAttributes」の「」を参照してください。 AWS SDK for PHP

Ruby

SDK for Ruby

> **Note**
>
> の詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスSetTopicAttributes」の「」を参照してください。 AWS SDK for Ruby

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> の詳細については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、SetTopicAttributesAWS 「 SDK for SAP ABAP API リファレンス」の「」を参照してください。

AWS CloudFormationを使用した配信ステータスのログ記録を設定する

`DeliveryStatusLogging` を使用して を設定するには AWS CloudFormation、JSON または YAML テンプレートを使用して AWS CloudFormation スタックを作成します。詳細については、

「 ユーザーガイド」の「 AWS::SNS::Topicリソースの DeliveryStatusLoggingプロパティ AWS CloudFormation 」を参照してください。以下は、Amazon SQS プロトコルのすべてのDeliveryStatusLogging属性を使用して新しいトピックを作成したり、既存のトピックを更新したりするための JSON および YAML の AWS CloudFormation テンプレートの例です。 Amazon SQS

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 メッセージ配信の再試行

Amazon SNS は、各配信プロトコルの配信ポリシーを定義します。配信ポリシーは、サーバー側のエラーが発生したとき (サブスクライブされたエンドポイントをホストするシステムが利用できなくなったとき) に Amazon SNS がメッセージの配信を再試行する方法を定義します。配信ポリシーが枯渇すると、Amazon SNS は配信の再試行を停止し、にデッドレターキューが添付されていない限り、メッセージを破棄します。詳細については、「Amazon SNS デッドレターキュー (DLQ)」を参照してください。

トピック

- 配信プロトコルとポリシー
- 配信ポリシーの段階
- HTTP/S 配信ポリシーの作成

配信プロトコルとポリシー

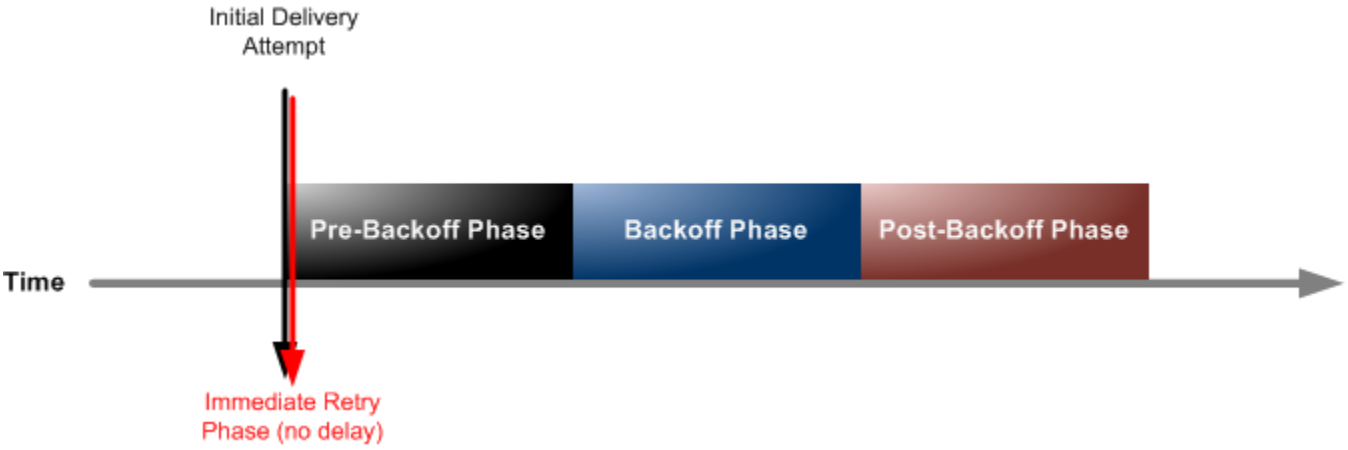
> **Note**
>
> - HTTP/S を除き、Amazon SNS で定義された配信ポリシーを変更することはできません。HTTP/S のみがカスタムポリシーをサポートしています。「HTTP/S 配信ポリシーの作成」を参照してください。
> - Amazon SNS は、配信の再試行にジッタリングを適用します。詳細については、『AWS アーキテクチャブログ』の「エクスポネンシャルバックオフとジッター」を参照してください。
> - HTTP/S エンドポイントのポリシー再試行時間の合計は 3,600 秒を超えることはできません。これはハード制限であり、時間を延長することはできません。

[エンドポイントタイプ]	配信プロトコル	即時の再試行 (遅延なし) 段階	バックオフ前段階	バックオフ段階	バックオフ後段階	合計試行回数
AWS 管理対象エンドポイント	Amazon データ Firehose [1]	3 回、遅延なし	2 回、1 秒間隔で	10 回、エクスポネンシャルバックオフで 1 秒から 20 秒まで	100,000 回、20 秒間隔で	100,015 回、23 日間
	AWS Lambda					
	Amazon SQS					
カスタマー管理のエンドポイント	SMTP	0 回、遅延なし	2 回、10 秒間隔で	10 回、エクスポネンシャルバックオフで 10 秒から 600 秒まで (10 分)	38 回、600 秒 (10 分) 間隔で	50 回、6 時間以上
	SMS					
	モバイルプッシュ					

[1] Firehose プロトコルによるスロットリングエラーについては、Amazon SNS はお客様が管理するエンドポイントと同じ配信ポリシーを使用します。

配信ポリシーの段階

次の図は、配信ポリシーの段階を示しています。



各配信ポリシーは、4 つの段階で構成されます。

1. 即時の再試行段階(遅延なし) - この段階は遅延なしの段階とも呼ばれ、最初の配信の試行の直後に発生します。この段階では再試行間の遅延時間はありません。
2. バックオフ前段階 - この段階は即時の再試行段階の後に続きます。Amazon SNS は、バックオフ関数を適用する前に、この段階を使用して一連の再試行を試みます。この段階では、再試行回数と再試行間の遅延量を指定します。
3. バックオフ段階 - この段階では、再試行バックオフ関数を使用して、再試行間の遅延を制御します。この段階では、最小遅延、最大遅延、および遅延が最小遅延から最大遅延までどれだけ速く増加するかを定義する再試行バックオフ関数を設定します。バックオフ関数は、数論的、指数、幾何学的、または一次です。
4. バックオフ後段階 - この段階はバックオフ段階の後に続きます。再試行回数とその間の遅延量を指定します。これが最終段階です。

HTTP/S 配信ポリシーの作成

配信ポリシーとその 4 つの段階を使用して、Amazon SNS が HTTP/S エンドポイントへのメッセージ配信を再試行する方法を定義できます。Amazon SNS では、HTTP サーバーの容量に基づいてポリシーをカスタマイズする場合など、HTTP エンドポイントのデフォルトの再試行ポリシーを上書きできます。

HTTP/S 配信ポリシーは、サブスクリプションレベルまたはトピックレベルで JSON オブジェクトとして設定できます。トピックレベルでポリシーを定義すると、そのトピックに関連付けられたすべての HTTP/S サブスクリプションに適用されます。配信ポリシーをサブスクリプションレベルで設定するには、`Subscribe` または `SetSubscriptionAttributes` API アクションのいずれかを使用できます。配信ポリシーをトピックレベルで設定するには、`CreateTopic` または

SetTopicAttributes API アクションのいずれかを使用できます。または、テンプレート内のリソースを使用することもできます。AWS::SNS::Subscription AWS CloudFormation

HTTP/S サーバーの容量に応じて配信ポリシーをカスタマイズする必要があります。ポリシーは、トピック属性またはサブスクリプション属性として設定できます。トピック内のすべての HTTP/S サブスクリプションが同じ HTTP/S サーバーをターゲットにする場合は、トピックのすべての HTTP/S サブスクリプションで有効になるように、配信ポリシーをトピック属性として設定することをお勧めします。それ以外の場合は、ポリシーがターゲットとする HTTP/S サーバーの容量に応じて、トピック内の HTTP/S サブスクリプションごとに配信ポリシーを作成する必要があります。

リクエストポリシーに Content-Type ヘッダーを設定して、通知のメディアタイプを指定することもできます。デフォルトでは、Amazon SNS は、コンテンツタイプが `text/plain; charset=UTF-8` に設定された HTTP/S エンドポイントにすべての通知を送信します。Amazon SNS では、デフォルトのリクエストポリシーを上書きできます。サポート対象の `headerContentType` と制約事項については、次の表を参照してください。

次の JSON オブジェクトは、失敗した HTTP/S 配信を再試行するように Amazon SNS に次のように指示する配信ポリシーを表します。

1. 遅延なし段階ですぐに 3 回
2. バックオフ前段階で 2 回 (1 秒間隔で)
3. 10 回 (1 秒から 60 秒までのエクスポネンシャルバックオフで)
4. バックオフ後段階で 35 回 (60 秒間隔で)

このサンプル配信ポリシーでは、Amazon SNS は合計 50 回試行してからメッセージを破棄します。配信ポリシーで指定された再試行を実行し尽くした後もメッセージを保持するには、配信不能メッセージをデッドレターキュー (DLQ) に移動するようにサブスクリプションを構成します。詳細については、「Amazon SNS デッドレターキュー (DLQ)」を参照してください。

> **Note**
>
> また、この配信ポリシーでは、`maxReceivesPerSecond` を使用して、配信を毎秒 10 以下に抑えるよう Amazon SNS に指示しています。このセルフスロットリングレートでは、配信されたメッセージ (アウトバウンドトラフィック) よりも公開されたメッセージ (インバウンドトラフィック) の方が多くなる可能性があります。インバウンドトラフィックがアウトバウンドトラフィックよりも多い場合、サブスクリプションによって大きなメッセージバックログが蓄積され、メッセージ配信のレイテンシーが高くなる可能性があります。配信

ポリシーでは、必ず maxReceivesPerSecond ワークロードに悪影響を与えない値を指定してください。

> **Note**
>
> この配信ポリシーは、application/json に対する HTTP/S 通知のデフォルトのコンテンツタイプをオーバーライドします。

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

配信ポリシーは、再試行ポリシー、スロットルポリシー、リクエストポリシーで構成されます。配信ポリシーには、合計で 9 つの属性があります。

ポリシー	説明	制約事項
minDelayTarget	再試行の最小遅延。 単位: 秒	1 から最大遅延まで デフォルト: 20
maxDelayTarget	再試行の最大遅延。 単位: 秒	最小遅延から 3,600 まで デフォルト: 20

ポリシー	説明	制約事項
numRetries	即時再試行、バックオフ前再試行、バックオフ再試行、ポストバックオフ再試行の合計数。	0 ~ 100 デフォルト: 3
numNoDelayRetries	即時に行う再試行の回数。再試行の間に遅延はありません。	0 以上 デフォルト: 0
numMinDelayRetries	バックオフ前段階での再試行回数と、これらの間に指定された最小遅延時間。	0 以上 デフォルト: 0
numMaxDelayRetries	バックオフ後段階での再試行回数と、その間の最大遅延。	0 以上 デフォルト: 0
backoffFunction	再試行間のバックオフのモデル。	次の 4 つのオプションのいずれか。 • 数論的 • 指数 • 幾何学的 • 一次 デフォルト: linear
maxReceivesPerSecond	サブスクリプションごとの 1 秒あたりの配信の最大数。	1 以上 デフォルト: スロットリングなし

ポリシー	説明	制約事項
`headerContentType`	HTTP/S エンドポイントに送信される通知のコンテンツタイプ。	リクエストポリシーが定義されていない場合、コンテンツタイプはデフォルトで `text/plain; charset=UTF-8` に設定されます。 サブスクリプションに対する raw メッセージ配信が無効になっている場合 (デフォルト)、または配信ポリシーがトピックレベルで定義されている場合、サポートされるヘッダーコンテンツタイプは `application/json` および `text/plain` です。 サブスクリプションに対する raw メッセージ配信を有効にすると、以下のコンテンツタイプがサポートされます。 • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • application/octet-stream • application/soap+xml • アプリケーション/ x-www-form-urlencoded • application/xhtml+xml

ポリシー	説明	制約事項
		• application/xml

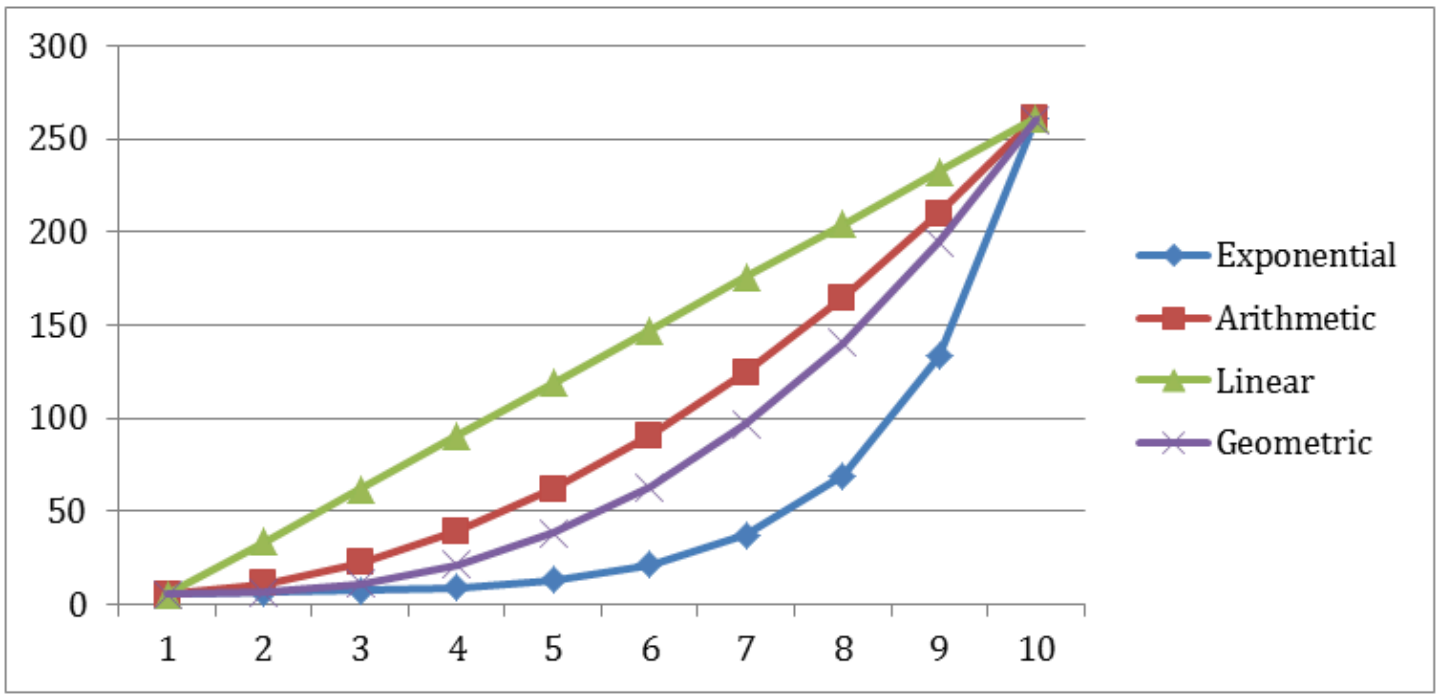
Amazon SNS は、次の式を使用して、バックオフ段階での再試行回数を計算します。

```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

3 つのパラメータを使用して、バックオフ段階での再試行の頻度を制御できます。

- `minDelayTarget` - バックオフ段階の最初の再試行に関連する遅延を定義します。
- `maxDelayTarget` - バックオフ段階の最後の再試行に関連する遅延を定義します。
- `backoffFunction` - バックオフ段階の最初の再試行と最後の再試行の間のすべての再試行に関連する遅延時間を計算するために Amazon SNS で使用されるアルゴリズムを定義します。4 つの再試行バックオフ関数の 1 つを使用できます。

次の図は、各再試行バックオフ関数が、バックオフ段階での再試行に関連する遅延にどのように影響するかを示しています。再試行の合計回数を 10、最小遅延を 5 秒、最大遅延を 260 秒に設定した配信ポリシーです。縦軸は、10 回ごとの再試行に関連する遅延時間を秒単位で表します。横軸は、最初の試行から 10 回目の試行までの再試行回数を表します。



Amazon SNS デッドレターキュー (DLQ)

デッドレターキューは、Amazon SNS サブスクリプションが受信者に正常に配信できないメッセージの送信先としての Amazon SQS キューです。クライアントエラーまたはサーバーエラーが原因で配信できないメッセージは、詳細な分析や再処理のためにデッドレターキューに保持されます。詳細については、サブスクリプションの Amazon SNS デッドレターキューを設定する および Amazon SNS メッセージ配信の再試行 を参照してください。

> **Note**
>
> - Amazon SNS サブスクリプションと Amazon SQS キューは、同じ AWS アカウントとリージョン内になければいけません。
> - FIFO トピックでは、Amazon SNS サブスクリプションのデッドレターキューとして Amazon SQS キューを使用できます。FIFO トピックサブスクリプションでは FIFO キューを使用します。標準トピックサブスクリプションでは標準キューを使用します。
> - 暗号化された Amazon SQS キューをデッドレターキューとして使用するには、Amazon SNS サービスプリンシパルに AWS KMS API アクションへのアクセス権を付与するキーポリシーのあるカスタム KMS を使用する必要があります。詳細については、このガイドの「保管中の暗号化」と、『Amazon Simple Queue Service デベロッパーガイド』の「サーバー側の暗号化 (SSE) と AWS KMS を使用した Amazon SQS データの保護」を参照してください。

トピック

メッセージ配信が失敗する理由

一般に、Amazon SNS がクライアント側またはサーバー側のエラーにより、サブスクライブされたエンドポイントにアクセスできない場合、メッセージの配信は失敗します。Amazon SNS がクライ

アント側のエラーを受信した場合、または対応する再試行ポリシーで指定された再試行回数を超えるメッセージに対してサーバー側のエラーを受信し続ける場合、Amazon SNS はメッセージを廃棄します。ただし、デッドレターキューがサブスクリプションに添付されている場合を除きます。配信に失敗しても、サブスクリプションのステータスは変更されません。詳細については、「Amazon SNS メッセージ配信の再試行」を参照してください。

クライアント側エラー

Amazon SNS に古いサブスクリプションのメタデータがあると、クライアント側のエラーが発生する可能性があります。これらのエラーは、通常、所有者がエンドポイント (Amazon SNS トピックにサブスクライブされている Lambda 関数など) を削除した場合や、サブスクライブしているエンドポイントに添付されているポリシーを、Amazon SNS がエンドポイントにメッセージを配信できないように所有者が変更した場合に発生します。Amazon SNS は、クライアント側のエラーの結果として失敗したメッセージ配信を再試行しません。

サーバー側のエラー

サーバー側のエラーは、サブスクライブされたエンドポイントを担当するシステムが利用できなくなったり、Amazon SNS からの有効なリクエストを処理できないことを示す例外を返す場合に発生します。サーバー側のエラーが発生すると、Amazon SNS は一次バックオフ関数またはエクスポネンシャルバックオフ関数を使用して、失敗した配信を再試行します。Amazon SQS または AWS Lambda によってバックアップされた AWS 管理のエンドポイントに起因するサーバー側エラーについては、Amazon SNS は 23 日間にわたって 100,015 回まで配信を再試行します。

カスタマー管理のエンドポイント (HTTP、SMTP、SMS、モバイルプッシュなど) も、サーバー側のエラーを引き起こす可能性があります。Amazon SNS は、これらのタイプのエンドポイントへの配信も再試行します。HTTP エンドポイントはお客様定義の再試行ポリシーをサポートしますが、Amazon SNS は SMTP、SMS、およびモバイルプッシュエンドポイントに対して、内部配信再試行ポリシーを 6 時間にわたって 50 回に設定します。

デッドレターキューのしくみ

メッセージの配信はサブスクリプションレベルで行われるため、デッドレターキューは (トピックではなく) Amazon SNS サブスクリプションに添付されます。これにより、各メッセージの元のターゲットエンドポイントをより簡単に識別できます。

Amazon SNS サブスクリプションに関連付けられたデッドレターキューは、通常の Amazon SQS キューです。メッセージの保持期間の詳細については、『Amazon Simple Queue Service デベロッパーガイド』の「メッセージに関連するクォータ」を参照してください。Amazon SQS

SetQueueAttributes API アクションを使用して、メッセージの保持期間を変更できます。アプリケーションの復元性を高めるために、デッドレターキューの最大保持期間を 14 日に設定することをお勧めします。

メッセージがデッドレターキューに移動する仕組み

メッセージは、再処理ポリシーを使用してデッドレターキューに移動されます。再処理ポリシーは、デッドレターキューの ARN を参照する JSON オブジェクトです。deadLetterTargetArn 属性は ARN を指定します。ARN は、Amazon SNS サブスクリプションと同じ AWS アカウント およびリージョンの Amazon SQS キューを指している必要があります。詳細については、「サブスクリプションの Amazon SNS デッドレターキューを設定する」を参照してください。

次の JSON オブジェクトは、SNS サブスクリプションに添付された再処理ポリシーのサンプルです。

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

メッセージをデッドレターキューから移動する方法

メッセージをデッドレターキューから移動するには、次の 2 つの方法があります。

- Amazon SQS コンシューマロジックの作成を避ける - デッドレターキューをイベントソースとして Lambda 関数に設定して、デッドレターキューを吸い出します。
- Amazon SQS コンシューマロジックを作成する - Amazon SQS API、AWS SDK、または AWS CLI を使用して、デッドレターキュー内のメッセージのポーリング、処理、削除を行うカスタムコンシューマーロジックを記述します。

デッドレターキューのモニタリングとログ記録方法

Amazon CloudWatch メトリクスを使用して、Amazon SNS サブスクリプションに関連付けられたデッドレターキューをモニタリングできます。すべての Amazon SQS キューは 1 分間隔で CloudWatch メトリクスを送信します。詳細については、『Amazon Simple Queue Service デベロッパーガイド』の「Amazon SQS で使用できる CloudWatch メトリクス」を参照してください。デッドレターキューを持つすべての Amazon SNS サブスクリプションも、CloudWatch メトリクスを出力します。詳細については、「CloudWatch を使用した Amazon SNS のモニタリング」を参照してください。

デッドレターキューのアクティビティを通知するには、CloudWatch メトリクスとアラームを使用できます。例えば、デッドレターキューが常に空であると予想される場合、NumberOfMessagesSent メトリクスの CloudWatch アラームを作成できます。アラームのしきい値を 0 に設定し、アラームが発動したときに通知する Amazon SNS トピックを指定できます。この Amazon SNS トピックは、任意のエンドポイントタイプ (E メールアドレス、電話番号、モバイルポケットベルアプリなど) にアラーム通知を配信できます。

CloudWatch Logs を使用して、Amazon SNS 配信が失敗する原因となる例外や、デッドレターキューに送信されるメッセージを調査できます。Amazon SNS は、配信の成功と失敗の両方を CloudWatch に記録できます。詳細については、「Amazon SNS メッセージ配信ステータス」を参照してください。

サブスクリプションの Amazon SNS デッドレターキューを設定する

デッドレターキューは、Amazon SNS サブスクリプションが受信者に正常に配信できないメッセージの送信先としての Amazon SQS キューです。クライアントエラーまたはサーバーエラーが原因で配信できないメッセージは、詳細な分析や再処理のためにデッドレターキューに保持されます。詳細については、「Amazon SNS デッドレターキュー (DLQ)」および「Amazon SNS メッセージ配信の再試行」を参照してください。

このページでは AWS Management Console、、、、、を使用して Amazon SNS サブスクリプションのデッドレターキューを設定する方法を説明します。 AWS AWS CLI AWS CloudFormation

> **Note**
>
> FIFO トピックでは、Amazon SNS サブスクリプションのデッドレターキューとして Amazon SQS キューを使用できます。FIFO トピックサブスクリプションでは FIFO キューを使用し、標準トピックサブスクリプションでは標準キューを使用します。

前提条件

デッドレターキューを設定するには、次の前提条件を満たしている必要があります。

1. MyTopic という名前で Amazon SNS トピックを作成する。
2. MyEndpoint という名前で Amazon SQS キューを作成し、Amazon SNS サブスクリプションのエンドポイントとして使用します。
3. (スキップ用 AWS CloudFormation) キューをトピックにサブスクライブします。

4. MyDeadLetterQueue という名前で Amazon SQS キューを作成し、Amazon SNS サブスクリプションのデッドレターキューとして使用します。
5. Amazon SQS API アクションへのアクセスを Amazon SNS プリンシパルに付与するには、MyDeadLetterQueue に次のキューポリシーを設定します。

```
{
  "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"
      }
    }
  }]
}
```

トピック

- を使用して Amazon SNS サブスクリプションのデッドレターキューを設定するには AWS Management Console
- SDK を使用して Amazon SNS サブスクリプションのデッドレターキューを設定するには AWS
- を使用して Amazon SNS サブスクリプションのデッドレターキューを設定するには AWS CLI
- を使用して Amazon SNS サブスクリプションのデッドレターキューを設定するには AWS CloudFormation

を使用して Amazon SNS サブスクリプションのデッドレターキューを設定するには AWS Management Console

このチュートリアルを開始する前に、次の前提条件を完了してください。

1. Amazon SQS コンソールにサインインします。
2. Amazon SQS キューを作成するか、既存のキューを使用して、キューの [詳細] タブでキューの ARN を確認します。次に例を示します。

```
arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue
```

3. Amazon SNS コンソールにサインインします。
4. ナビゲーションパネルで、[サブスクリプション] を選択します。
5. [サブスクリプション] ページで、既存のサブスクリプションを選択し、[編集] を選択します。
6. [Edit ***1234a567-bc89-012d-3e45-6fg7h890123i***] ページで、[再処理ポリシー (デッドレターキュー)] セクションを展開し、次の操作を行います。

   a. [有効] を選択します。

   b. Amazon SQS キューの ARN を指定します。
7. [変更を保存] をクリックします。

   デッドレターキューを使用するようにサブスクリプションが設定されます。

SDK を使用して Amazon SNS サブスクリプションのデッドレターキューを設定するには AWS

この例を実行する前に、必ずの前提条件を完了してください。

AWS SDK を使用するには、認証情報を使用して設定する必要があります。詳細については、「AWS SDK とツールのリファレンスガイド」の「共有設定ファイルと認証情報ファイル」を参照してください。

次のコード例は使用方法を示していますSetSubscriptionAttributesRedrivePolicy。

Java

SDK for Java 1.x

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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);
```

を使用して Amazon SNS サブスクリプションのデッドレターキューを設定するには AWS CLI

このチュートリアルを開始する前に、次の前提条件を完了してください。

1. AWS CLIをインストールして設定します。詳細については、「AWS Command Line Interface ユーザーガイド」を参照してください。
2. 以下のコマンドを使用します。

```
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\"}"
```

を使用して Amazon SNS サブスクリプションのデッドレターキューを設定するには AWS CloudFormation

このチュートリアルを開始する前に、次の前提条件を完了してください。

1. 次のJSON コードをMyDeadLetterQueue.jsonという名前のファイルにコピーします。

```
{
  "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. AWS CloudFormation コンソール にサインインします。
3. [テンプレートの選択] ページで、[テンプレートを Amazon S3 にアップロード] を選択した後、MyDeadLetterQueue.json ファイルを選択し、次に [次へ] を選びます。
4. [詳細の指定] ページで、[スタック名] に MyDeadLetterQueue を入力してから、[次へ] を選択します。
5. [オプション] ページで、[次へ] を選択します。
6. [Review] ページで、[作成] を選択します。

   AWS CloudFormation MyDeadLetterQueueスタックの作成を開始し、CREATE_IN_PROGRESS ステータスを表示します。プロセスが完了すると、CREATE_COMPLETE AWS CloudFormation ステータスが表示されます。

Amazon SNS メッセージのアーカイブ、リプレイ、および分析

Amazon SNS 標準トピックは、Amazon Data Firehose を介したメッセージアーカイブをサポートしています。Firehose 配信ストリームに通知をファンアウトできます。これにより、Amazon Simple Storage Service (Amazon S3)、Amazon Redshift など、Firehose がサポートするストレージと分析の宛先に通知を送信できます。

Amazon SNS FIFO トピックは、トピック所有者がトピックに発行されたメッセージを最大 365 日間保存 (またはアーカイブ) できる、ノーコードのインプレースメッセージアーカイブをサポートしています。アクティブ `ArchivePolicy` を使用するトピックの場合、サブスクライバーは `ReplayPolicy` を作成して、アーカイブされたメッセージをサブスクライブされたエンドポイントに対してリトリーブ (またはリプレイ) できます。この機能の詳細は、「FIFO トピックのメッセージのアーカイブとリプレイ」を参照してください。

機能	標準トピック	FIFO トピック
メッセージのアーカイブ	Firehose 配信ストリームへのファンアウト	FIFO トピック所有者のメッセージアーカイブ
メッセージのリプレイ	標準トピックのリプレイは組み込み機能ではありません。多くのお客様は、メッセージアーカイブに基づいて独自の機能を構築しています。	FIFO トピックサブスクライバーのメッセージリプレイ

Amazon SNS を使用した Application-to-Application (A2A) メッセージング

このセクションでは、受信者とのアプリケーショントゥーアプリケーションのメッセージングのためのAmazon SNS の使用方法について説明します。

トピック

Firehose 配信ストリームへのファンアウト

Amazon Data Firehose 配信ストリームを Amazon Amazon SNSトピックにサブスクライブできます。これにより、追加のストレージおよび分析エンドポイントに通知を送信できます。Amazon SNS トピックに発行されたメッセージは、サブスクライブされた Firehose 配信ストリームに送信され、Firehose で設定された送信先に配信されます。サブスクリプション所有者は、Amazon SNS トピックに最大 5 つの Firehose 配信ストリームをサブスクライブできます。各 Firehose 配信ストリームには、リクエストと 1 秒あたりのスループットのデフォルトクォータがあります。この制限により、メッセージの配信数 (アウトバウンドトラフィック) よりもメッセージの発行数 (インバウンドトラフィック) が多くなる場合があります。インバウンドトラフィックがアウトバウンドトラフィックよりも多いと、サブスクリプションによって大きなメッセージバックログが蓄積され、メッセージ配信のレイテンシーが高くなる場合があります。ワークロードへの悪影響を避けるため、発行レートに基づいてクォータの引き上げをリクエストできます。

Firehose 配信ストリームを通じて、Amazon SNS 通知を Amazon Simple Storage Service (Amazon S3)、Amazon Redshift、Amazon OpenSearch Service (OpenSearch サービス) 、および Datadog、New Relic、MongoDB、Splunk などのサードパーティーサービスプロバイダーにファンアウトできます。

例えば、この機能を使用して、コンプライアンス、アーカイブ、またはその他の目的で Amazon S3 バケットにトピックに送信されたメッセージを永続的に保存できます。これを行うには、S3 バケッ

トの送信先を持つ Firehose 配信ストリームを作成し、その配信ストリームを Amazon SNS トピックにサブスクライブします。別の例として、Amazon SNS トピックに送信されたメッセージの分析を実行するには、 OpenSearch サービスインデックスの送信先を使用して配信ストリームを作成します。その後、Firehose 配信ストリームを Amazon SNS トピックにサブスクライブできます。

Amazon SNS は、Firehose エンドポイントに送信される通知のメッセージ配信ステータスのログ記録もサポートしています。詳細については、「Amazon SNS メッセージ配信ステータス」を参照してください。

トピック

- Firehose 配信ストリームを Amazon SNS トピックにサブスクライブするための前提条件
- Firehose 配信ストリームを Amazon SNS トピックにサブスクライブする
- 配信ストリームの送信先を使用する
- メッセージのアーカイブと分析のユースケースの例

Firehose 配信ストリームを Amazon SNS トピックにサブスクライブするための前提条件

Amazon Data Firehose 配信ストリームを SNS トピックにサブスクライブするには、 に以下AWS アカウントが必要です。

- スタンダード SNS トピック 詳細については、「Amazon SNS トピックを作成する」を参照してください。
- Firehose 配信ストリーム。詳細については、「Amazon Data Firehose デベロッパーガイド」の「Amazon Data Firehose 配信ストリームの作成」および「Firehose リソースへのアクセス権をアプリケーションに付与する」を参照してください。
- Amazon SNS サービスプリンシパルを信頼し、配信ストリームへの書き込み権限を持つ AWS Identity and Access Management (IAM) ロール サブスクリプションを作成するときに、このロールの Amazon リソースネーム (ARN) を `SubscriptionRoleARN` として入力します。Amazon SNS はこのロールを引き受けます。これにより、Amazon SNS は Firehose 配信ストリームにレコードを配置できます。

  次のポリシーの例は、推奨されるアクセス権限を示しています。

```
{
  "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"
    }
  ]
}
```

Firehose を使用するための完全なアクセス許可を付与するには、 AWS管理ポリシー を使用することもできますAmazonKinesisFirehoseFullAccess。または、Firehose を使用するためのより厳格なアクセス許可を提供するには、独自のポリシーを作成します。少なくとも、ポリシーは、特定の配信ストリー ムで PutRecord オペレーションを実行するアクセス権限を与える必要があります。

いずれの場合も、信頼関係を編集して、Amazon SNS サービスプリンシパルを含める必要があります。例:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

ロールの作成について詳しくは、『IAM ユーザーガイド』の「AWS のサービスに許可を委任するロールの作成」を参照してください。

これらの要件を満たしたら、配信ストリームを SNS トピックへサブスクライブできます。

Firehose 配信ストリームを Amazon SNS トピックにサブスクライブする

Amazon SNS 通知を Amazon Data Firehose 配信ストリームに配信するには、まずすべての前提条件 を満たしていることを確認してください。サポートされているエンドポイントのリストについては、「」の「Amazon Data Firehose エンドポイントとクォータ」を参照してくださいAmazon Web Services 全般のリファレンス。

Firehose 配信ストリームをトピックにサブスクライブするには

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションペインで [サブスクリプション] を選択します。
3. [サブスクリプション] ページで [サブスクリプションの作成] を選択します。
4. [サブスクリプションの作成] ページで [詳細] セクションで、以下を実行します。
   a. [トピック ARN] は、標準トピックの Amazon リソースネーム (ARN) を選択します。
   b. プロトコル で、Firehose を選択します。
   c. エンドポイント で、Amazon SNS から通知を受信できる Firehose 配信ストリームの ARN を選択します。
   d. サブスクリプションロール ARN には、Firehose 配信ストリームへの書き込み用に作成した AWS Identity and Access Management (IAM) ロールの ARN を指定します。詳細については、「Firehose 配信ストリームを Amazon SNS トピックにサブスクライブするための前提条件」を参照してください。
   e. (オプション) 発行されたメッセージから Amazon SNS メタデータを削除するには、[Enable raw message delivery] を選択します。詳細については、「Amazon SNS raw メッセージの配信」を参照してください。
5. (オプション) フィルターポリシーを設定するには、[サブスクリプションのフィルターポリシー] セクションを展開します。詳細については、「Amazon SNS サブスクリプションフィルターポリシー」を参照してください。
6. (オプション) サブスクリプションのデッドレターキューを設定するには、Redrive ポリシー (デッドレターキュー)を展開します。詳細については、「Amazon SNS デッドレターキュー (DLQ)」を参照してください。
7. [サブスクリプションの作成] を選択します。

コンソールがサブスクリプションを作成し、サブスクリプションの [詳細] ページを開きます。

配信ストリームの送信先を使用する

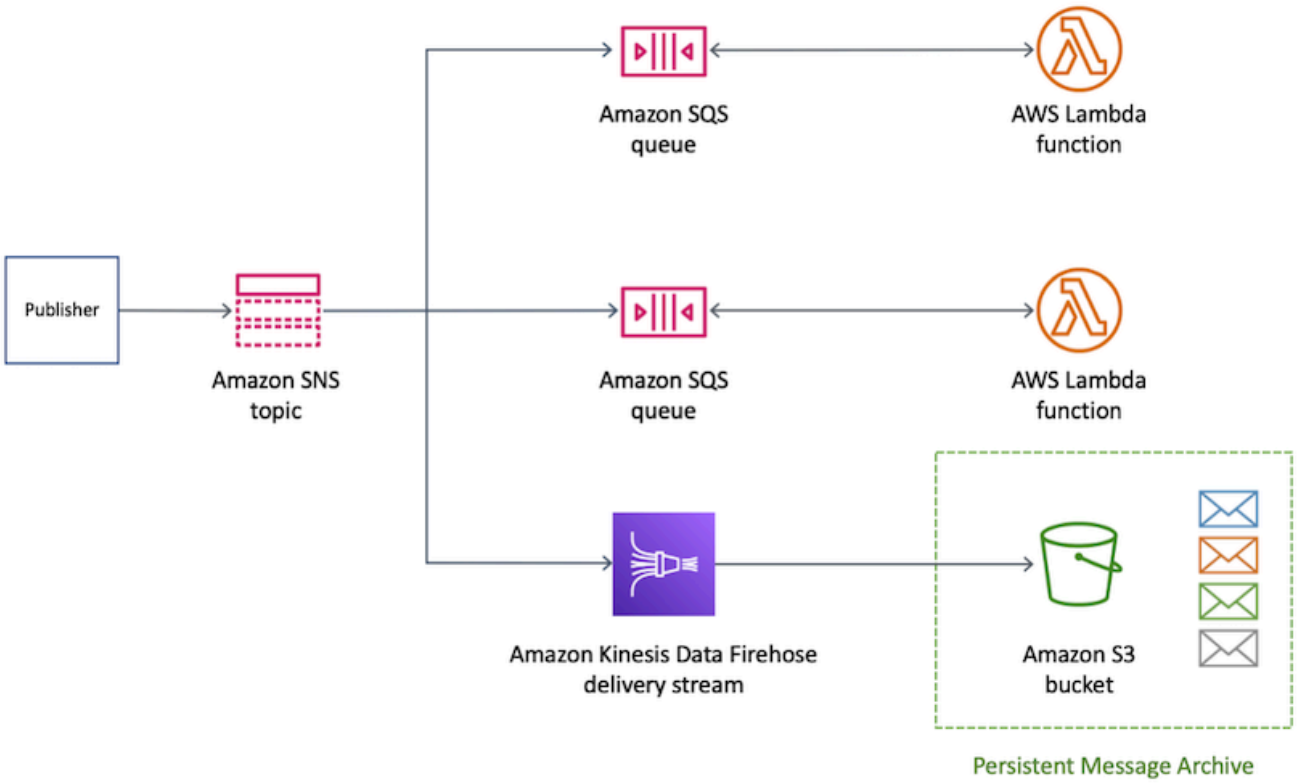
Amazon Data Firehose 配信ストリームを介して、追加のエンドポイントにメッセージを送信できます。このセクションでは、サポートされている送信先の操作方法について説明します。

トピック

- Amazon S3 の送信先
- OpenSearch サービスの送信先
- Amazon Redshift の送信先
- HTTP 送信先

Amazon S3 の送信先

このセクションでは、Amazon Simple Storage Service (Amazon S3) にデータを発行する Amazon Data Firehose 配信ストリームについて説明します。



トピック

- Amazon S3 の送信先のアーカイブされたメッセージ形式

- Amazon S3 送信先のメッセージを分析する

Amazon S3 の送信先のアーカイブされたメッセージ形式

次の例は、読みやすくするためにインデントを使用して Amazon Simple Storage Service (Amazon S3) バケットに送信された Amazon SNS 通知を示しています。

> **Note**
>
> この例では、raw メッセージ配信は、発行されたメッセージに対して無効になっています。raw メッセージ配信が無効になっている場合、Amazon SNS は次のプロパティを含む JSON メタデータをメッセージに追加します。
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> raw 配信の詳細については、「Amazon SNS raw メッセージの配信」を参照してください。

```
{
    "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"
        }
    }
}
```

次の例は、Amazon Data Firehose 配信ストリームを介して同じ Amazon S3 バケットに送信される 3 つの Amazon S3を示しています。バッファリングが考慮され、改行によってメッセージが分離されます。

```
{"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"}
```

Amazon S3 送信先のメッセージを分析する

このページでは、Amazon Data Firehose 配信ストリームを介して Amazon Simple Storage Service (Amazon S3) の送信先に送信された Amazon SNS Amazon S3メッセージを分析する方法について説明します。

Firehose 配信ストリームを介して Amazon S3 の送信先に送信された SNSメッセージを分析するには

1. Amazon S3 リソースを設定します。手順については、『Amazon Simple Storage Service ユーザーガイド』の「バケットの作成」および『Amazon Simple Storage Service ユーザーガイド』の「Amazon S3 バケットの使用」を参照してください。
2. 配信ストリームを設定します。手順については、「Amazon Data Firehose デベロッパーガイド」の「送信先に Amazon S3 を選択する」を参照してください。
3. Amazon Athena を使用して、標準 SQL を使用した Amazon S3 オブジェクトのクエリを実行します。詳細については、『Amazon Athena ユーザーガイド』の「開始方法」を参照してください。

クエリの例

このクエリの例では、次のことを前提としています。

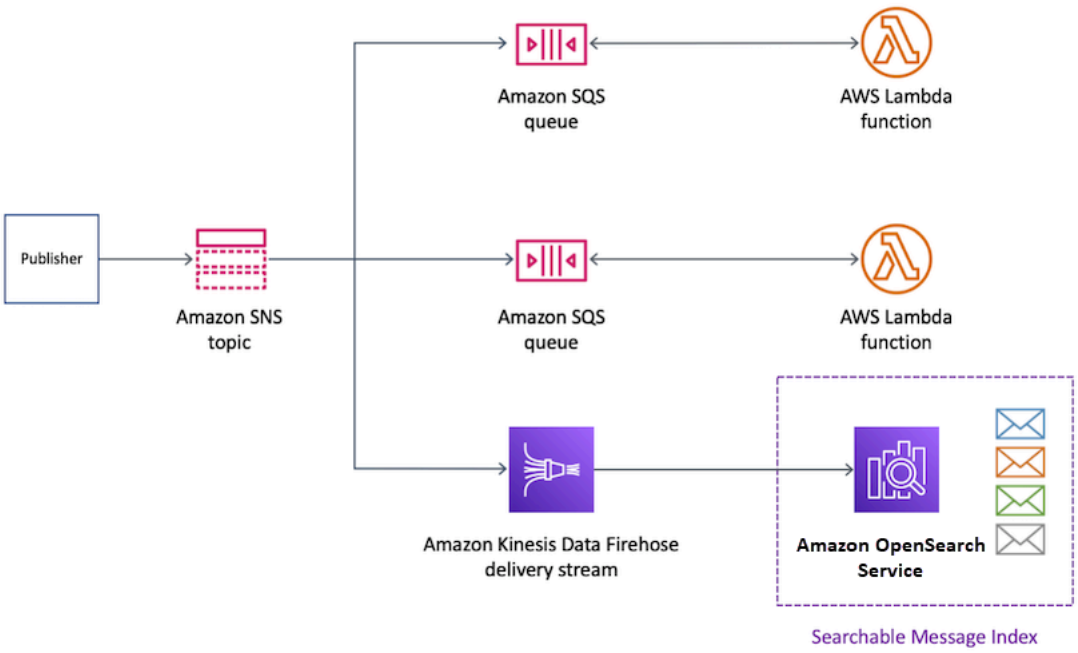
- メッセージは `default` スキーマの `notifications` テーブルに保存されます。
- `notifications` テーブルは `string` のタイプの `timestamp` カラムを含みます。

次のクエリは、指定された日付の範囲内で受信されたすべてのSNS メッセージを返します。

```
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';
```

OpenSearch サービスの送信先

このセクションでは、Amazon OpenSearch Service (OpenSearch Service) にデータを発行する Amazon Data Firehose 配信ストリームについて説明します。



トピック

- OpenSearch Service インデックスのアーカイブされたメッセージ形式
- OpenSearch サービス送信先のメッセージの分析

OpenSearch Service インデックスのアーカイブされたメッセージ形式

次の例は、`my-index` という名前の Amazon OpenSearch Service (OpenSearch Service) インデックスに送信された Amazon SNS 通知を示しています。このインデックスには、`Timestamp` フィールドにタイムフィルターフィールドがあります。SNS 通知はペイロードの `_source` プロパティにあります。

> **Note**
>
> この例では、raw メッセージ配信は、発行されたメッセージに対して無効になっています。raw メッセージ配信が無効になっている場合、Amazon SNS は次のプロパティを含む JSON メタデータをメッセージに追加します。
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`

- Timestamp
- UnsubscribeURL
- MessageAttributes

raw 配信の詳細については、「Amazon SNS raw メッセージの配信」を参照してください。


```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "49613100963111323203250405402193283794773886550985932802.0",
  "_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
  ]
}
```

OpenSearch サービス送信先のメッセージの分析

このページでは、Amazon Data Firehose 配信ストリームを介して Amazon Service (Service) 送信先に送信される Amazon SNS メッセージを分析する方法について説明します。 OpenSearch OpenSearch

Firehose 配信ストリームを介して OpenSearch サービス送信先に送信された SNS メッセージを分析するには

1. OpenSearch サービスリソースを設定します。手順については、「Amazon OpenSearch Service デベロッパーガイド」の「Amazon OpenSearch Service の開始方法」を参照してください。
2. 配信ストリームを設定します。手順については、「Amazon Data Firehose デベロッパーガイド」の「送信先に OpenSearch サービスを選択する」を参照してください。
3. OpenSearch サービスクエリと Kibana を使用してクエリを実行します。詳細については、「Amazon OpenSearch Service デベロッパーガイド」の「ステップ 3: OpenSearch サービスドメインでドキュメントを検索する」および「Kibana」を参照してください。

クエリの例

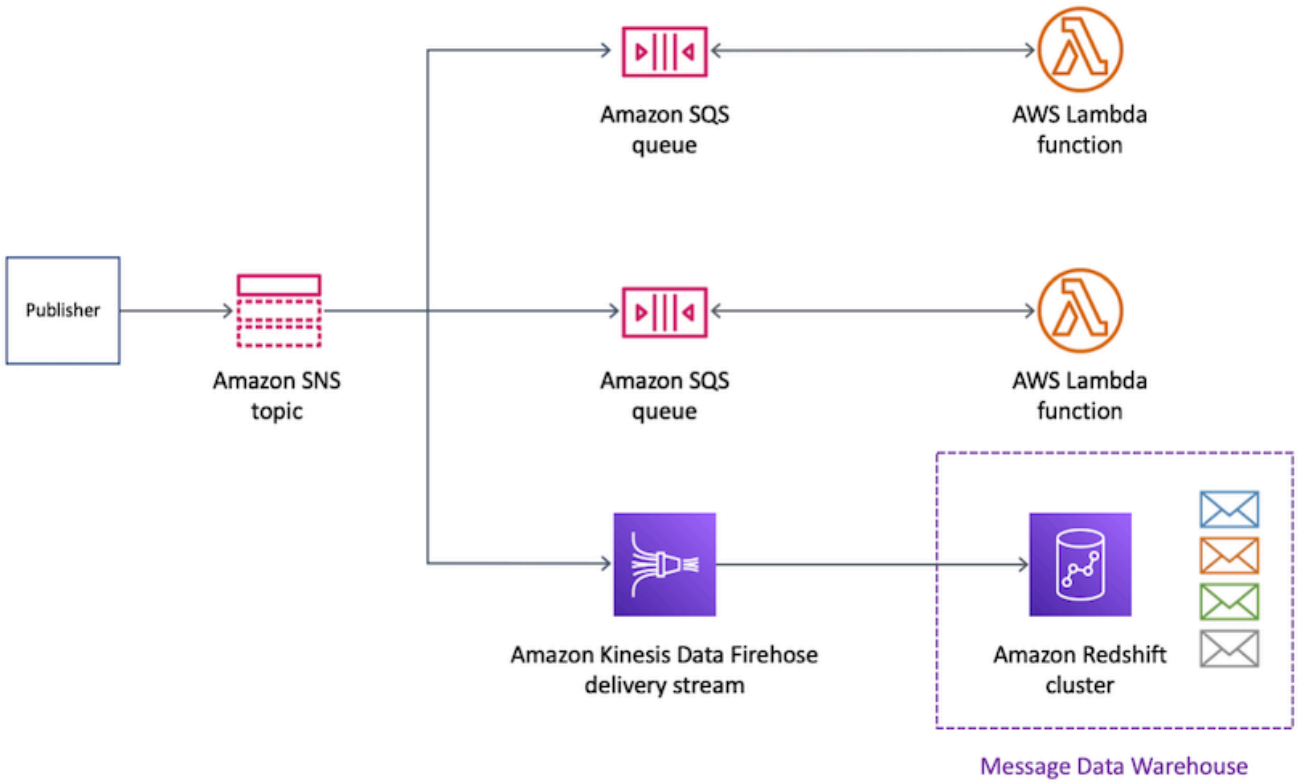
次の例では、指定された日付の範囲内で受信したすべての SNS メッセージの `my-index` インデックスのクエリを示しています。

```
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"
            }
          }
        }
      ]
    }
  }
```

```
}
```

Amazon Redshift の送信先

このセクションでは、Amazon Redshift にデータを発行する Amazon Data Firehose 配信ストリームに Amazon SNS 通知をファンアウトする方法について説明します。この設定で、Amazon Redshift データベースに接続し、このデータベースで特定の条件を満たす Amazon SNS メッセージを SQL クエリツールを使用してクエリできます。



トピック

- Amazon Redshift 送信先のアーカイブテーブル構造
- Amazon Redshift 送信先のメッセージを分析する

Amazon Redshift 送信先のアーカイブテーブル構造

Amazon Redshift エンドポイントの場合、パブリッシュされた Amazon SNS メッセージは、テーブル内の列としてアーカイブされます。次に例を示します。

> **Note**
>
> この例では、raw メッセージ配信は、発行されたメッセージに対して無効になっています。raw メッセージ配信が無効になっている場合、Amazon SNS は次のプロパティを含む JSON メタデータをメッセージに追加します。
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Message`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> raw 配信の詳細については、「Amazon SNS raw メッセージの配信」を参照してください。Amazon SNS は、このリストに表示される頭文字を大文字の単語でメッセージにプロパティを追加しますが、Amazon Redshift テーブルではすべて小文字で表示されます。Amazon Redshift エンドポイントの JSON メタデータを変換するには、SQL の `COPY` コマンドを使用できます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「JSON からのコピーの例」および「'auto ignorecase' オプションを使用した JSON データからのロード」を参照してください。

type	messageId	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
通知	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	サンプル件名	サンプルメッセージ	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscrib	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageId	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
						e&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	

type	messageId	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
Notification	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	サンプル件名 2	サンプルメッセージ 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageId	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
Notification	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	サンプル件名 3	サンプルメッセージ 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Amazon Redshift エンドポイントへの通知のファンアウトの詳細については、「Amazon Redshift の送信先」を参照してください。

Amazon Redshift 送信先のメッセージを分析する

このページでは、Amazon Data Firehose 配信ストリームを介して Amazon Redshift の送信先に送信される Amazon SNS メッセージを分析する方法について説明します。

Firehose 配信ストリームを介して Amazon Redshift の送信先に送信された SNS メッセージを分析するには

1. Amazon Redshift リソースを設定します。手順については、『Amazon Redshift 使用開始ガイド』の「Amazon Redshift の開始方法を参照してください。
2. 配信ストリームを設定します。手順については、「Amazon Data Firehose デベロッパーガイド」の「送信先に Amazon Redshift を選択する」を参照してください。
3. クエリを実行する 詳細については、「Amazon Redshift 管理ガイド」の「クエリエディタを使用してデータベースのクエリを実行する」を参照してください。

クエリの例

このクエリの例では、次のことを前提としています。

- メッセージはデフォルト `public` スキーマの `notifications` テーブルに保存されます。
- SNS メッセージからの `Timestamp` プロパティは、`timestamptz` のタイプのコラムデータを持つテーブルの `timestamp` カラムに保存されます。

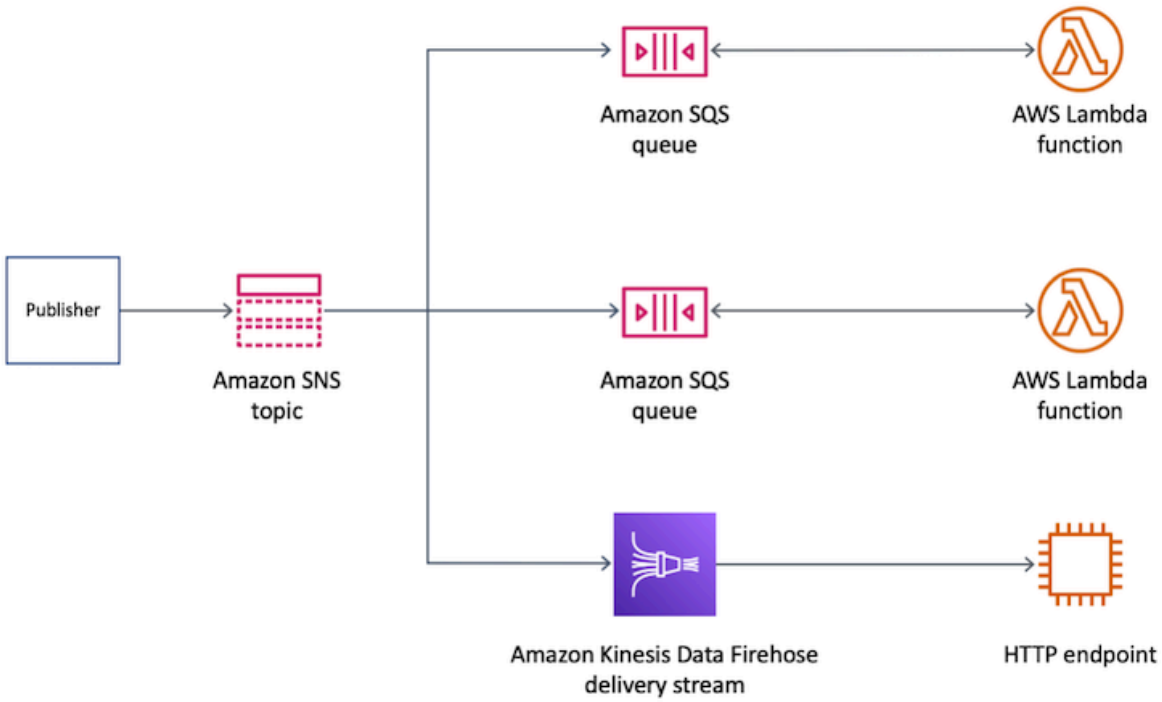
> **Note**
>
> Amazon Redshift エンドポイントの JSON メタデータを変換するには、SQL `COPY` コマンドを使用できます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「JSON からのコピーの例」および「'auto ignorecase' オプションを使用した JSON データからのロード」を参照してください。

次のクエリは、指定された日付の範囲内で受信されたすべてのSNS メッセージを返します。

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

HTTP 送信先

このセクションでは、HTTP エンドポイントにデータを発行する Amazon Data Firehose 配信ストリームについて説明します。



トピック

- HTTP 送信先の配信メッセージ形式

HTTP 送信先の配信メッセージ形式

以下は、Amazon Data Firehose 配信ストリームが HTTP エンドポイントに送信できる Amazon SNS からの HTTP POST リクエストボディの例です。SNS 通知は、`records` プロパティで base64 ペイロードとしてエンコードされます。

> **Note**
>
> この例では、raw メッセージ配信は、発行されたメッセージに対して無効になっています。raw 配信の詳細については、「Amazon SNS raw メッセージの配信」を参照してください。

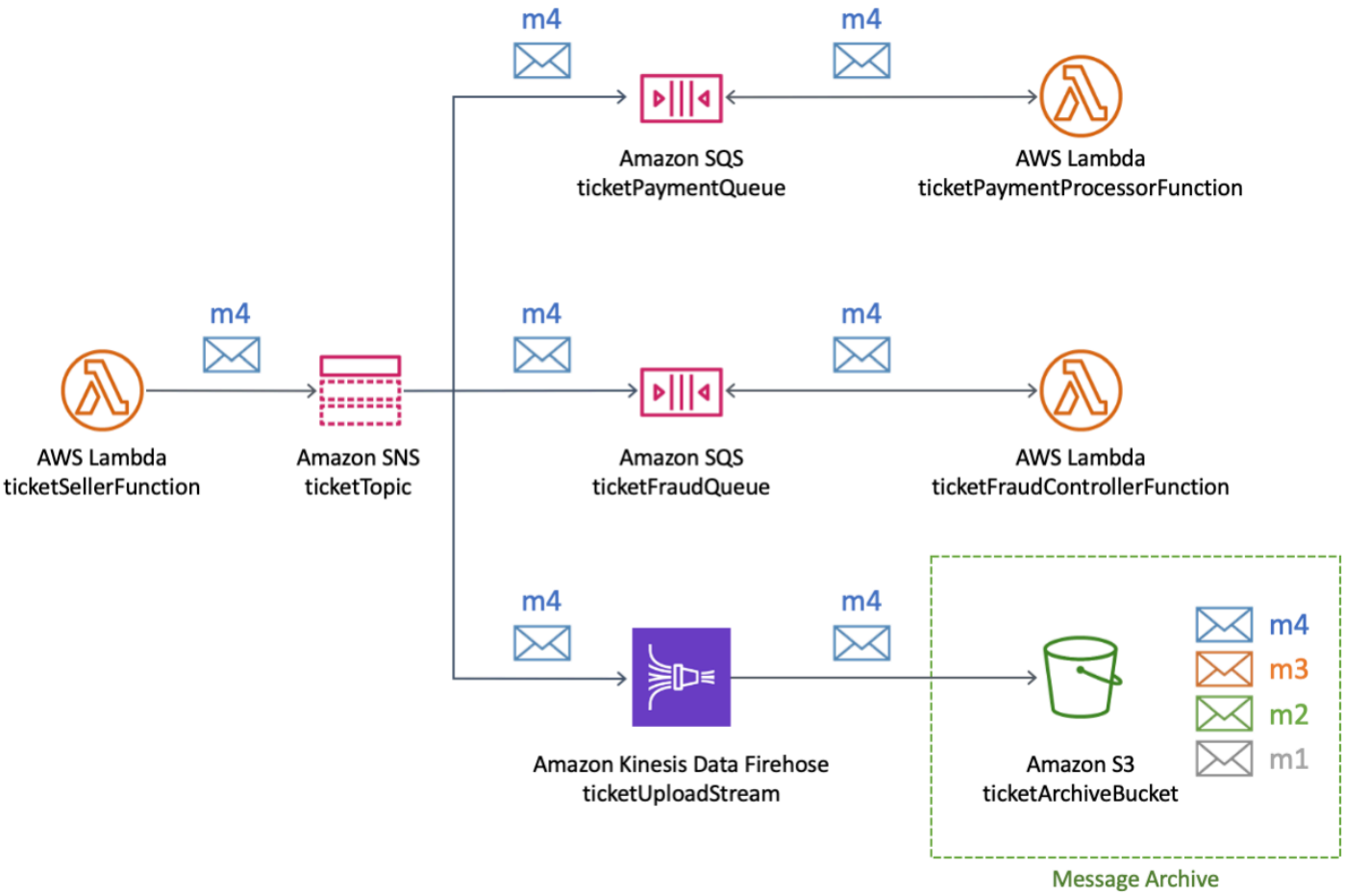
```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
```

```
    "records": [
      {
        "data":
"eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```

メッセージのアーカイブと分析のユースケースの例

このセクションでは、Amazon SNS メッセージをアーカイブおよび分析するための一般的なユースケースのチュートリアルを行います。

このユースケースの設定は、規制された環境で動作する航空会社の発券プラットフォームです。このプラットフォームは、少なくとも5年間、すべてのチケット販売をアーカイブする必要があるコンプライアンスフレームワークの対象となります。データ保持に関するコンプライアンス目標を達成するために、企業は Amazon Data Firehose 配信ストリームを既存の SNS トピックにサブスクライブします。配信ストリームの送信先は、Amazon Simple Storage Service (Amazon S3) バケットです。この設定では、SNS トピックに発行されたすべてのイベントが Amazon S3 バケットにアーカイブされます。次の図は、この設定のアーキテクチャを示しています。



分析を実行し、チケット販売に関する洞察を得るために、同社は Amazon Athena を使用して SQL クエリを実行します。例えば、同社は、最も人気のある目的地や最も頻繁に搭乗する人についてクエリを実行できます。

このユースケースで AWS リソースを作成するには、AWS Management Console または AWS CloudFormation テンプレートを使用できます。

トピック

- 初期リソースを作成する
- Firehose 配信ストリームの作成
- Firehose 配信ストリームを Amazon SNS トピックにサブスクライブする
- 構成のテストとクエリを実行する
- AWS CloudFormation テンプレートを使用する

初期リソースを作成する

このページでは、メッセージのアーカイブと分析例のユースケースのリソースを作成する方法について説明します。

- 1 つの Amazon Simple Storage Service (Amazon S3) バケット
- 2 つの Amazon Simple Queue Service (Amazon SQS) キュー
- 1 つの Amazon SNS トピック
- Amazon SNS トピックへの 2 つの Amazon SQS キューサブスクリプション

初期リソースを作成するには

1. Amazon S3 バケットを作成する

   a. Amazon S3 コンソールを開きます。
   b. [バケットを作成する] を選択します。
   c. [バケット名] に、一意の名前を入力します。他のフィールドはデフォルトのままにします。
   d. [バケットの作成] をクリックします。

   Amazon S3 バケットについては、『Amazon Simple Storage Service ユーザーガイド』の「バケットの作成」および『Amazon Simple Storage Service ユーザーガイド』の「Amazon S3 バケットの使用」を参照してください。
2. 2 つの Amazon SQS キューを作成します。

   a. Amazon SQS コンソールを開きます。
   b. [キューの作成] を選択します。
   c. タイプ] で、[標準] を選択します。
   d. [名前] に **ticketPaymentQueue** と入力します。
   e. [アクセスポリシー] の [Choose method] で、[Advanced] を選択します。
   f. [JSON ポリシー] ボックスに、以下のポリシーをペーストします。

```
{
  "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"
        }
      }
    }
  ]
}
```

このアクセスポリシーでは、AWS アカウント 番号 (*123456789012*) を自分で変更し、それに応じて AWS リージョン (*us-east-1*) を変更します。

g. [キューの作成] を選択します。

h. 上記のステップを繰り返して、**ticketFraudQueue** という 2 番目の SQS キューを作成します。

SQS キューの作成の詳細については、『Amazon Simple Queue Service デベロッパーガイド』の「Amazon SQS キューの作成 (コンソール)」を参照してください。

3. SNS トピックを作成する

a. Amazon SNS コンソールの [トピック] ページを開きます。

b. [トピックの作成] を選択します。

c. [詳細] の [タイプ] で、[標準] を選択します。

d. 名前 に **ticketTopic** と入力します。

e. [トピックの作成] を選択します。

SNS トピックの作成の詳細については、「Amazon SNS トピックを作成する」を参照してください。

4. SQS キューを SNS トピックにサブスクライブする

a. Amazon SNS コンソールで、[ticketTopic] トピック詳細ページの [サブスクリプションの作成] を選択します。

b. [詳細] で、[プロトコル] は [Amazon SQS] を選択します。

c. [エンドポイント] で、ticketPaymentQueue キューの Amazon リソースネーム (ARN) を選択します。

d. [Create subscription] を選択します。

e. 上記のステップを繰り返して、ticketFraudQueue キューの ARN を使用して 2 番目のサブスクリプションを作成します。

SNS トピックへのサブスクライブに関する詳細については、「Amazon SNS トピックへサブスクライブする」を参照してください。また、Amazon SQS コンソールから SNS トピックに SQS キューをサブスクライブすることもできます。詳細については、『Amazon Simple Queue Service デベロッパーガイド』の「Amazon SNS トピックへの Amazon SQS キューのサブスクライブ (コンソール)」を参照してください。

これで、このユースケース例の初期リソースが作成できました。続行するには、「Firehose 配信ストリームの作成」を参照してください。

Firehose 配信ストリームの作成

このページでは、メッセージのアーカイブと分析の例のユースケース 用に Amazon Data Firehose 配信ストリームを作成する方法について説明します。

Firehose 配信ストリームを作成するには

1. Amazon Kinesis サービスコンソールを開きます。
2. Firehose を選択し、配信ストリームの作成 を選択します。
3. [New delivery stream] ページで、[Delivery stream name] に、**`ticketUploadStream`** を入力して、次に [Next] をクリックします。
4. [Recipients] ページで、[Next] をクリックします。
5. [Choose a Destination] ページで、次の作業を行います。

   a. 送信先で、[Amazon S3] を選択します。

   b. [S3 送信先] の [S3 バケット] で、最初に作成した S3 バケットを選択します。

   c. [Next] をクリックします。
6. [設定] ページの [S3 buffer conditions] で、次の作業を行います。

   - [Buffer size] に **`1`** を入力します。
   - [Buffer interval] に **`60`** を入力します。

Amazon S3 バッファにこれらの値を使用すると、設定をすばやくテストできます。最初に満たされたバッファ条件によって、S3 バケットへのデータの送信がトリガーされます。

7. [設定] ページの [Permissions] を選択すると AWS Identity and Access Management (IAM) ロールの作成に必要なアクセス権限が自動的に割り当てられます。続いて、[Next] をクリックします。
8. [Review] ページで、[配信ストリームの作成] を選択します。。
9. Kinesis Data Firehose 配信ストリームページから、先ほど作成した配信ストリームを選択します (ticketUploadStream) 。[詳細] タブで、後でストリームの Amazon リソースネーム (ARN) を記録します。

配信ストリームの作成の詳細については、「Amazon Data Firehose デベロッパーガイド」の「Amazon Data Firehose 配信ストリームの作成」を参照してください。 IAM ロールの作成について詳しくは、『IAM ユーザーガイド』の「AWS のサービスに許可を委任するロールの作成」を参照してください。

必要なアクセス許可を持つ Firehose 配信ストリームが作成されました。続行するには、「Firehose 配信ストリームを Amazon SNS トピックにサブスクライブする」を参照してください。

Firehose 配信ストリームを Amazon SNS トピックにサブスクライブする

このページでは、メッセージのアーカイブと分析のユースケース例向けの以下を作成する方法について説明します。

- Amazon SNS サブスクリプションが Amazon Data Firehose 配信ストリームにレコードを配置できるようにする AWS Identity and Access Management (IAM) ロール
- SNS トピックへの Firehose 配信ストリームサブスクリプション

Amazon SNS サブスクリプションの IAM ロールを作成するには

1. IAM コンソールの [ロール] ページを開きます。
2. [ロールの作成] を選択します。
3. [信頼されたエンティティのタイプの選択)] で、[AWS のサービス] を選択します。
4. [ユースケースの選択] で、[SNS] を選択します。接ぎ木、[Next: Permissions] を選択します。
5. [次へ: タグ] を選択します。
6. [次へ: レビュー] を選択します。

7. [Review] ページの [Role name] に、**ticketUploadStreamSubscriptionRole** を入力します。次に、[ロールの作成] を選択します。
8. ロールが作成されたら、その名前 () を選択しますticketUploadStreamSubscriptionRole。
9. ロールの [Summary] ページで、[Add inline policy] を選択します。
10. [ポリシーの作成] ページの [JSON] タブを選択し、次のポリシーをテキストボックスにペーストします。

```
{
    "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"
        }
    ]
}
```

このポリシーでは、AWS アカウント 番号 (*123456789012*) を自分で変更し、それに応じて AWS リージョン (*us-east-1*) を変更します。

11. [ポリシーの確認] を選択します。
12. [ポリシーの確認] ページで [名前] に **FirehoseSnsPolicy** と入力します。次に、[ポリシーの作成] を選択します。
13. ロールの [Summary] ページで、後で [ロール ARN] を記録します。

IAM ロールの作成について詳しくは、『IAM ユーザーガイド』の「AWS のサービスに許可を委任するロールの作成」を参照してください。

Firehose 配信ストリームを SNS トピックにサブスクライブするには

1. Amazon SNS コンソールの [トピック] ページを開きます。
2. [サブスクリプション] タブで [サブスクリプションの作成] を選択します。
3. 詳細 で、プロトコル で Amazon Data Firehose を選択します。
4. エンドポイント には、前に作成したticketUploadStream配信ストリームの Amazon リソースネーム (ARN) を入力します。たとえば、**arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream** と入力します。
5. サブスクリプションロール ARN には、前に作成した IAM ticketUploadStreamSubscriptionRole ロールの ARN を入力します。たとえば、**arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole** と入力します。
6. [Enable raw message delivery] チェックボックスをオンにします。
7. [サブスクリプションを作成] を選択します。

これで、IAM ロールと SNS トピックのサブスクリプションを作成しました。続行するには、「構成のテストとクエリを実行する」を参照してください。

構成のテストとクエリを実行する

このページでは、Amazon SNS トピックにメッセージを発行することにより、メッセージのアーカイブと分析例のユースケースをテストする方法について説明します。手順には、実行してお客様独自のニーズに適応できるクエリの例が含まれています。

設定をテストするには

1. Amazon SNS コンソールの [トピック] ページを開きます。
2. [**ticketTopic** トピック] を選択します。
3. [メッセージの発行] を選択します。
4. [Publish message to topic] ページで、メッセージ本文に次のように入力します。メッセージの末尾に改行文字を追加します。

   ```
   {"BookingDate":"2020-12-15","BookingTime":"2020-12-15
    04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
   ```

   他のすべてのオプションはデフォルト設定のままにします。
5. [メッセージの発行] を選択します。

公開の詳細については、「Amazon SNS メッセージの発行」を参照してください。

6. 配信ストリームの間隔が 60 秒になったら、はじめに作成した Amazon Simple Storage Service (Amazon S3) コンソールと Amazon S3 バケットを選択します。

   バケットには、発行されたメッセージが表示されます。

データをクエリするには

1. Amazon Athena コンソールを開きます。
2. クエリを実行する

   例えば、default スキーマの notifications テーブルには、以下のデータが含まれていることを前提とします。

```
{"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"}
```

   トップの送信先を確認するには、以下のクエリを実行します。

```
SELECT destination
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

   特定の日付と時刻の範囲内で販売されたチケットをクエリするには、以下のようなクエリを実行します。

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
```

```
  AND TIMESTAMP '2020-12-15 12:00:00';
```

両方のサンプルクエリをお客様独自のニーズに合わせて調整できます。Athena を使用してクエリを実行する方法の詳細については、『Amazon Athena ユーザーガイド』の「開始方法」を参照してください。

クリーンアップ

テスト完了後に利用料金が発生しないようにするには、チュートリアル中に作成した以下のリソースを削除します。

- Amazon SNS サブスクリプション
- Amazon SNS トピック
- Amazon Simple Queue Service Amazon SQSキュー
- Amazon S3 バケット
- Amazon Data Firehose 配信ストリーム
- AWS Identity and Access Management AM ロールとポリシー

AWS CloudFormation テンプレートを使用する

以下の YAML テンプレートを使用して、メッセージのアーカイブと分析例のユースケースの Amazon SNS のデプロイを自動化できます。

```
---
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
```

Lambda 関数へのファンアウト

Amazon SNS および AWS Lambda は、Lambda 関数を Amazon SNS 通知で呼び出すことができるように統合されています。Lambda 関数がサブスクライブしている SNS トピックにメッセージが発行されると、発行されたメッセージのペイロードで Lambda 関数が呼び出されます。Lambda 関数は、メッセージペイロードを入力パラメータとして受け取り、メッセージの情報の操作、他の SNS トピックへのメッセージの発行、または他の AWS サービスへのメッセージの送信を行うことができます。

さらに、Amazon SNS は、Lambda エンドポイントに送信されたメッセージ通知のメッセージ配信ステータス属性もサポートします。詳細については、「Amazon SNS メッセージ配信ステータス」を参照してください。

Prerequisites

Amazon SNS 通知を使用して Lambda 関数を呼び出すには、以下が必要になります。

- Lambda 関数
- Amazon SNS トピック

Amazon SNS で使用する Lambda 関数の作成の詳細については、「Using Lambda with Amazon SNS」を参照してください。Amazon SNS トピックの作成の詳細については、「トピックの作成」を参照してください。

Amazon SNS を使用してオプトインリージョンからデフォルトで有効になっているリージョンにメッセージを配信する場合、プリンシパル `sns.amazonaws.com` を `sns.<opt-in-`

region>.amazonaws.com に置き換えて、AWS Lambda 関数で作成されたポリシーを変更する必要があります。

例えば、米国東部 (バージニア北部) の Lambda 関数をアジアパシフィック (香港) の SNS トピックにサブスクライブする場合は、AWS Lambda 関数ポリシーのプリンシパルを sns.ap-east-1.amazonaws.com に変更します。オプトインリージョンには、2019 年 3 月 20 日以降に開始されたリージョンが含まれます。これには、アジアパシフィック (香港)、中東 (バーレーン)、欧州 (ミラノ)、アフリカ (ケープタウン) が含まれます。2019 年 3 月 20 日以前に開始されたリージョンは、デフォルトで有効になっています。

> **Note**
>
> AWS は、デフォルトで有効になっているリージョンからオプトインリージョンへの Lambda へのクロスリージョン配信はサポートしていません。オプトインリージョンから他のオプトインリージョンへの SNS メッセージのクロスリージョン転送もサポートされていません。

関数をトピックへサブスクライブする

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. [トピック] ページでトピックを選択します。
4. [サブスクリプション] セクションで、[サブスクリプションの作成] を選択します。
5. [サブスクリプションの作成] ページの [詳細] セクションで、以下を実行します。
   a. 選択した [トピック ARN] を確認します。
   b. [プロトコル] で AWS Lambda を選択します。
   c. [エンドポイント] に関数の ARN を入力します。
   d. [サブスクリプションの作成] を選択します。

Lambda 関数がサブスクライブしている SNS トピックにメッセージが発行されると、発行されたメッセージのペイロードで Lambda 関数が呼び出されます。チュートリアルを含む、AWS Lambda での Amazon SNS の使用方法の詳細については、「Using AWS Lambda with Amazon SNS」を参照してください。

Amazon SQS キューへのファンアウト

Amazon SNS は、Amazon Simple Queue Service (Amazon SQS) と密接に連携します。これらのサービスは、デベロッパーにとって異なる利点を提供します。Amazon SNS を使用すると、アプリケーションは「プッシュ」メカニズムを使用して、複数の受信者にタイムクリティカルなメッセージを送信することができます。そのため、更新を定期的に確認または「ポーリング」する必要がありません。Amazon SQS は、ポーリングモデルを通してメッセージを交換するために分散型アプリケーションによって使用されるメッセージキューサービスであり、このサービスを使用すると、送信および受信コンポーネントを切り離すことができます。この場合、各コンポーネントが同時に使用できなくなることはありません。Amazon SNS と Amazon SQS を組み合わせて使用することで、イベントの即時通知を必要とするアプリケーションにメッセージを配信できるだけでなく、他のアプリケーションのメッセージは、後で処理できるように Amazon SQS キューに保持することもできます。

Amazon SNS トピックへの Amazon SQS キューをサブスクライブすると、そのトピックにメッセージを発行することができ、Amazon SNS は Amazon SQS メッセージをサブスクライブされたキューに送信します。Amazon SQS メッセージには、トピックに発行された件名とメッセージに加え、JSON ドキュメントのメッセージに関するメタデータが含まれます。Amazon SQS メッセージは、以下の JSON ドキュメントのようになります。

```
{
   "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"
}
```

Amazon SNS トピックへ Amazon SQS キューをサブスクライブする

Amazon SNS トピックから Amazon SQS キューにメッセージを送信できるようにするには、次のいずれかを実行します。

- プロセスを簡素化するため、Amazon SQS コンソールを使用します。詳細については、『Amazon Simple Queue Service デベロッパーガイド』の「Amazon SNS トピックへの Amazon SQS キューのサブスクライブ」を参照してください。
- 以下の手順に従います。
  1. メッセージの送信先にするキューとキューをサブスクライブするトピックの Amazon Resource Name (ARN) を取得します。
  2. Amazon SNS トピックに sqs:SendMessage 許可を付与し、キューにメッセージを送信できるようにします。
  3. キューを Amazon SNS トピックへサブスクライブします。
  4. IAM ユーザーまたは AWS アカウント に、Amazon SNS トピックに発行し、Amazon SQS キューからのメッセージを読むための許可を付与します。
  5. トピックにメッセージを発行し、キューからのメッセージを読むことでテストします。

別の AWS アカウントにあるキューにメッセージを送信するようにトピックをセットアップする方法については、「別のアカウントの Amazon SQS キューへ Amazon SNS メッセージを送信する」を参照してください。

2 つのキューにメッセージを送信するトピックを作成する AWS CloudFormation テンプレートについては、「AWS CloudFormation テンプレートを使用して Amazon SQS キューにメッセージを送信するトピックを作成する」を参照してください。

ステップ 1: キューとトピックの ARN を取得する

トピックにキューをサブスクライブするときは、キューの ARN のコピーが必要です。同様に、トピックがキューにメッセージを送ることを許可するには、トピックの ARN のコピーが必要です。

キューの ARN を取得するには、Amazon SQS コンソールまたは GetQueueAttributes API アクションを使用できます。

キューの ARN を Amazon SQS コンソールから取得するには

1. AWS Management Console にサインインし、Amazon SQS コンソール (https://console.aws.amazon.com/sqs/) を開きます。
2. ARN を取得するキューのチェックボックスをオンにします。
3. [詳細] タブから、Amazon SNS トピックへのサブスクライブに使用できるように ARN 値をコピーします。

トピックの ARN を取得するには、Amazon SNS コンソール、`sns-get-topic-attributes` コマンド、または `GetQueueAttributes` API アクションを使用できます。

トピックの ARN を Amazon SNS コンソールから取得するには

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、ARN を取得するトピックを選択します。
3. 詳細セクションで、ARN 値を使用して Amazon SNS トピックがメッセージを送信する許可を取得できるようにします。

ステップ 2: Amazon SQS キューにメッセージを送信する許可を Amazon SNS トピックに付与する

Amazon SNS トピックがキューにメッセージを送信できるようにするには、Amazon SNS トピックに `sqs:SendMessage` アクションの実行を許可するようにキューのポリシーを設定する必要があります。

トピックにキューをサブスクライブする前に、トピックとキューが必要です。トピックやキューをまだ作成していない場合は、ここで作成します。詳細については、「トピックの作成」と、「Amazon Simple Queue Service デベロッパーガイド」の「キューの作成」を参照してください。

キューにポリシーを設定するには、Amazon SQS コンソールまたは SetQueueAttributes API アクションを使用できます。開始する前に、キューにメッセージを送信できるようにするトピックの ARN があることを確認してください。複数のトピックにキューをサブスクライブする場合、ポリシーにはトピックごとに 1 つの `Statement` 要素が含まれている必要があります。

Amazon SQS コンソールを使用してキューに SendMessage ポリシーを設定するには

1. AWS Management Console にサインインし、Amazon SQS コンソール (https://console.aws.amazon.com/sqs/) を開きます。
2. ポリシーを設定するキューのチェックボックスをオンにし、[アクセスポリシー] タブを選択してから、[編集] を選択します。
3. [アクセスポリシー] セクションで、キューにアクセスできるユーザーを定義します。

   - トピックにアクションを許可する条件を追加します。
   - 以下の例に示すように、Amazon SNS サービスを `Principal` に設定します。
   - `aws:SourceArn` または `aws:SourceAccount` グローバル条件キーを使用して、混乱した代理シナリオから保護します。これらの条件キーを使用するには、値をトピックの

ARN に設定します。キューを複数のトピックにサブスクライブしている場合は、代わりに aws:SourceAccount を使用できます。

このポリシーは、MyTopic が MyQueue にメッセージを送ることを許可します。

```
{
  "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"
        }
      }
    }
  ]
}
```

ステップ 3: キューを Amazon SNS トピックへサブスクライブします。

トピックを介してキューにメッセージを送信するには、キューを Amazon SNS トピックにサブスクライブする必要があります。キューは ARN で指定します。トピックにサブスクライブするには、Amazon SNS コンソール、sns-subscribe CLI コマンド、または Subscribe API アクションを使用できます。開始する前に、サブスクライブするキューの ARN があることを確認してください。

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. [トピック] ページでトピックを選択します。
4. [*MyTopic*] ページの [サブスクリプション] ページで [サブスクリプションの作成] を選択します。
5. [サブスクリプションの作成] ページの [詳細] セクションで、以下を実行します。

a. [トピックの ARN] を検証します。
b. [プロトコル] で、[Amazon SQS] を選択します。
c. [エンドポイント] に、 Amazon SQS キューの ARN を入力します。
d. [サブスクリプションの作成] を選択します。

サブスクリプションが確認されると、新しいサブスクリプションの [サブスクリプション ID] にサブスクリプション ID が表示されます。キューの所有者がサブスクリプションを作成する場合、サブスクリプションは自動的に確認され、ほぼ瞬時にアクティブになります。

通常は、自分のアカウントの自分のトピックに自分のキューをサブスクライブします。ただし、別のアカウントのキューを自分のトピックにサブスクライブすることもできます。サブスクリプションを作成するユーザーがキューの所有者ではない場合 (例えば、アカウント A のユーザーがアカウント A のトピックにアカウント B のキューをサブスクライブする場合)、サブスクリプションの確認が必要です。別のアカウントのキューをサブスクライブし、サブスクリプションを確認する方法の詳細については、「別のアカウントの Amazon SQS キューへ Amazon SNS メッセージを送信する」を参照してください。

ステップ 4: 適切なトピックおよびキューアクションに対するアクセス権限をユーザーに付与する

適切なユーザーのみに Amazon SNS トピックへのメッセージの発行および Amazon SQS キューからのメッセージの読み取り/削除を許可するには、AWS Identity and Access Management (IAM) を使用する必要があります。IAM ユーザーに対するトピックおよびキューに対するアクションの制御の詳細については、「Amazon SNS でのアイデンティティベースのポリシーを使用する」および『Amazon Simple Queue Service デベロッパーガイド』の「Amazon SQS での Identity and Access Management」を参照してください。

トピックまたはキューへのアクセスは、以下の 2 つの方法で制御します。

- IAM ユーザーまたはグループにポリシーを追加する。ユーザーにトピックやキューへのアクセス権限を付与する最も簡単な方法として、グループを作成し、そのグループに適切なポリシーとユーザーを追加することができます。個々のユーザーに設定するポリシーを継続的に追跡するよりも、グループに対してユーザーを追加または削除する方がはるかに簡単です。
- トピックまたはキューにポリシーを追加する。別の AWS アカウントにトピックへのアクセス権限を追加する場合、そのプリンシパルとして、アクセス権限を付与する AWS アカウント を持っているポリシーを追加することが唯一の方法です。

ほとんどの場合は、最初の方法 (ポリシーをグループに適用し、適切なユーザーをグループに追加または削除することでアクセス権限を管理する) を使用します。別のアカウントのユーザーにアクセス権限を付与する場合は、2 番目の方法を使用する必要があります。

IAM ユーザーまたはグループにポリシーを追加する

IAM ユーザーまたはグループに次のポリシーを追加した場合、そのユーザーまたはそのグループのメンバーに、MyTopic トピックで sns:Publish アクションを実行する許可が付与されます。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

IAM ユーザーまたはグループに次のポリシーを追加した場合、ユーザーまたはそのグループのメンバーに、キューの「MyQueue 1」と「MyQueu 2」に対して sqs:ReceiveMessage および sqs:DeleteMessage アクションを実行する許可が付与されます。

```
{
  "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"
      ]
    }
  ]
}
```

トピックまたはキューにポリシーを追加する

以下のサンプルポリシーは、トピックとキューに別のアカウントのアクセス権限を付与する方法を示しています。

> **Note**
>
> アカウントのリソースへの別の AWS アカウント アクセス権限を付与する場合、管理者レベルのアクセス (ワイルドカードアクセス) の権限を持っている IAM ユーザーにも、そのリソースへのアクセス許可が付与されます。他のアカウントの他のすべての IAM ユーザーは、自動的にリソースへのアクセスが拒否されます。その AWS アカウント アクセスの特定の IAM ユーザーにリソースへのアクセス権を付与する場合、管理者レベルアクセス権を持っているアカウントまたは IAM ユーザーは、そのリソースのアクセス権限をそれらの IAM ユーザーに委任する必要があります。クロスアカウントの委任の詳細については、『IAM ガイドの使用』の「Enabling Cross-Account Access」を参照してください。

アカウント 123456789012 の「マイトピック」トピックに次のポリシーを追加した場合、そのトピックで sns:Publish アクションを実行するアクセス権限をアカウント 111122223333 に付与したことになります。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

アカウント 123456789012 のキュー「マイキュー」に以下のポリシーを追加した場合、そのキューで sqs:DeleteMessage および sqs:ReceiveMessage アクションを実行する許可をアカウント 111122223333 に付与したことになります。

```
{
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

ステップ 5: トピックのキューサブスクリプションをテストする

トピックのキューサブスクリプションは、トピックに発行し、トピックがキューに送信したメッセージを表示することでテストできます。

Amazon SNS コンソールを使用してトピックに発行するには

1. トピックに発行するアクセス権限を持っている AWS アカウント または IAM ユーザーの認証情報を使用して、AWS Management Console にサインインし、Amazon SNS コンソール (https://console.aws.amazon.com/sns/) を開きます。
2. ナビゲーションパネルで、トピックを選択し、[トピックに発行] を選択します。
3. [件名] ボックスに件名 (「**`Testing publish to queue`**」など) を入力し、[メッセージ] ボックスに任意のテキスト (「**`Hello world!`**」など) を入力して、[メッセージの発行] を選択します。「Your message has been successfully published.」というメッセージが表示されます。

Amazon SQS コンソールを使用してトピックからのメッセージを表示するには

1. キュー内のメッセージを表示するアクセス権限を持っている AWS アカウント または IAM ユーザーの認証情報を使用して、AWS Management Console にサインインし、Amazon SQS コンソール (https://console.aws.amazon.com/sqs/) を開きます。
2. トピックにサブスクライブされている [queue] (キュー) を選択します。
3. [Send and receive messages] (メッセージの送受信) を選択し、[Poll for messages] (メッセージのポーリング) を選択します。タイプが [通知] のメッセージが表示されます。

4. [本文] カラムで、[詳細] を選択します。[メッセージ詳細] ボックスに、トピックに発行した件名とメッセージを含む JSON ドキュメントが表示されます。メッセージは、以下の JSON ドキュメントのようになります。

```
{
  "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. [閉じる] を選択します。キューに通知メッセージを送信するトピックへの発行は、正常に終了しました。

AWS CloudFormation テンプレートを使用して Amazon SQS キューにメッセージを送信するトピックを作成する

AWS CloudFormation では、テンプレートファイルを使用して、AWS リソースの集合を単一のユニットとして作成/制御することができます。このセクションでは、キューに発行するトピックのデプロイを容易にするサンプルテンプレートを使用します。このテンプレートは、2 つのキューの作成、キューにサブスクリプションを持つトピックの作成、トピックがキューにメッセージを送信できるようにするポリシーのキューへの追加、これらのリソースへのアクセスを制御する IAM ユーザーとグループの作成を行って、セットアップ手順を代行します。

AWS CloudFormation テンプレートを使用した AWS リソースのデプロイの詳細については、『AWS CloudFormation ユーザーガイド』の「開始方法」を参照してください。

AWS CloudFormation テンプレートを使用して AWS アカウント 内のトピックとキューをセットアップする

このサンプルテンプレートでは、1 つは IAM グループのメンバーがトピックに発行するため、もう 1 つはキューからのメッセージを読み取るために、それぞれ適切なアクセス権限を付与された 2 つの Amazon SQS キューにメッセージを送信できる Amazon SNS トピックを作成します。このテンプレートは、各グループに追加される IAM ユーザーも作成します。

テンプレートの内容をファイルにコピーします。[AWS CloudFormation テンプレート] ページからテンプレートをダウンロードすることもできます。テンプレートページで [Browse sample templates by AWS service] 、[Amazon Simple Queue Service] の順に選択します。

MySNSTopic は 2 つのサブスクライブされたエンドポイント (MyQueue1 と MyQueue2 という 2 つの Amazon SQS キュー) に発行するようにセットアップされます。MyPublishTopicGroup は、メンバーが API アクションの [Publish] または [sns-publish] コマンドを使用して MySNSTopic に発行する許可を持つ IAM グループです。テンプレートは、MyPublishUser と MyQueueUser という IAM ユーザーを作成し、このユーザーにログインプロファイルとアクセスキーを付与します。このテンプレートを使用してスタックを作成するユーザーは、入力パラメータとしてログインプロファイル用のパスワードを指定します。テンプレートは、2 人の IAM ユーザー用に MyPublishUserKey と MyQueueUserKey というアクセスキーを作成します。AddUserToMyPublishTopicGroup は MyPublishTopicGroup に MyPublishUser を追加して、グループに割り当てられたアクセス権限がユーザーに付与されるようにします。

MyRDMessageQueueGroup はメンバーが ReceiveMessage および DeleteMessage API アクションを使用して、2 つの Amazon SQS キューからメッセージを読み取り/削除するためのアクセス権限を持つIAM グループです。AddUserToMyQueueGroup は MyQueueUser を MyRDMessageQueueGroup に追加して、グループに割り当てられたアクセス権限がユーザーに付与されるようにします。MyQueuePolicy は、MySNSTopic が 2 つのキューに通知を発行する許可を割り当てます。

以下のリストは AWS CloudFormation テンプレートの内容を示しています。

```
{
  "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"]
```

```
            }
          ]
        ]
      }
    }
  }
}
```

HTTP(S) エンドポイントへのファンアウト

Amazon SNS を使用して、1 つ以上の HTTP または HTTPS エンドポイントに通知メッセージを送信できます。エンドポイントをトピックにサブスクライブすると、トピックに通知を発行することができ、Amazon SNS は HTTP POST リクエストを送信し、通知の内容をサブスクライブしたエンドポイントに配信します。エンドポイントをサブスクライブする際は、Amazon SNS が POST リクエストをエンドポイントに送信するときに HTTP を使用するか HTTPS を使用するかを選択します。HTTPS を使用する場合は、以下に対して Amazon SNS のサポートを利用できます。

- Server Name Indication (SNI) - これにより、Amazon SNS は複数のドメインをホストするために複数の証明書を必要とするサーバーなど、SNI を必要とする HTTPS エンドポイントをサポートすることができます。SNI の詳細については、「Server Name Indication」を参照してください。

- Basic and Digest Access Authentication - これにより、HTTP POST リクエストで HTTPS URL にユーザーネームとパスワードを指定できます (`https://user:password@domain.com`、`https://user@domain.com` など)。HTTPS を使用する場合、ユーザーネームとパスワードは確立された SSL 接続で暗号化されます。ドメイン名のみがプレーンテキストで送信されます。Basic and Digest Access Authentication の詳細については、RFC-2617 を参照してください。

> **Important**
>
> Amazon SNS は現在、プライベート HTTP(S) エンドポイントをサポートしていません。HTTPS URL は API アクセスを許可したプリンシパルの場合、Amazon SNS `GetSubscriptionAttributes` API アクションからのみ取得できます。

> **Note**
>
> クライアントサービスは HTTP/1.1 401 Unauthorized ヘッダーレスポンスをサポートできる必要があります。

リクエストには、JSON ドキュメントの通知に関するメタデータとともに、トピックに発行された件名とメッセージが含まれます。リクエストは以下の HTTP POST リクエストのようになります。HTTP ヘッダーおよびリクエストボディの JSON 形式の詳細については、「HTTP/HTTPS ヘッダー」および「HTTP/HTTPS 通知の JSON 形式」を参照してください。

```
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"
}
```

トピック

トピック HTTP/S にエンドポイントをサブスクライブする

このセクションのページでは、HTTP/S エンドポイントを Amazon SNS トピックにサブスクライブする方法について説明します。

トピック

ステップ 1: エンドポイントで Amazon SNS メッセージを処理する準備が完了していることを確認する

HTTP または HTTPS エンドポイントをトピックにサブスクライブする前に、Amazon SNS がサブスクライブプションの確認および通知メッセージの送信に使用する HTTP POST のリクエストを処理する能力が HTTP または HTTPS エンドポイントにあることを確認する必要があります。通常、これは Amazon SNS からの HTTP リクエストを処理するウェブアプリケーション (例えば、エンドポイントホストが Apache と Tomcat で Linux を実行している場合は Java servlet) の作成とデプロイを意味します。HTTP エンドポイントをサブスクライブする場合、Amazon SNS はサブスクリプションの確認リクエストを送信します。サブスクリプションを作成するときに、エンドポイントはこのリクエストを受信して処理する準備ができている必要があります。Amazon SNS はこの時点でこのリクエストを送信するためです。Amazon SNS は、サブスクリプションが確認されるまでエンドポイントにメッセージを送信しません。Amazon SNS は、サブスクリプションを確認すると、サブスクライブしたトピックで発行アクションが実行されたときにエンドポイントに通知を送信します。

サブスクリプションの確認および通知メッセージを処理するようにエンドポイントを設定するには

1. コードは、Amazon SNS がエンドポイントに送信する HTTP POST リクエストの HTTP ヘッダーを読み取る必要があります。また、コードは Amazon SNS が送信したメッセージのタイプを示すヘッダーフィールド `x-amz-sns-message-type` を探す必要があります。ヘッダーを確認すると、HTTP リクエストの本文を解析することなく、メッセージタイプを判断できます。処理する必要がある 2 つのタイプ (`SubscriptionConfirmation` および `Notification`) があります。`UnsubscribeConfirmation` メッセージは、サブスクリプションがトピックから削除された場合のみ使用されます。

   HTTP ヘッダーの詳細については、「HTTP/HTTPS ヘッダー」を参照してください。以下の HTTP POST のリクエストはサブスクリプションの確認メッセージの例です。

```
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. コードは、Amazon SNS メッセージを構成する名前と値のペアを読み取る HTTP POST リクエストの本文と content-type text/plain で、JSON ドキュメントを解析する必要があります。制御文字のエスケープ表現を ASCII 文字値に戻す変換 (\n を改行文字に変換するなど) を実行する JSON パーサーを使用します。Jackson JSON プロセッサなどの既存の JSON パーサーを使用するか、または独自のパーサーを記述できます。件名フィールドとメッセージフィールドで有効な JSON としてテキストを送信するためには、Amazon SNS は一部の制御文字を、JSON ドキュメントに含めることができるエスケープ表現に変換する必要があります。エンドポイントに送信される POST リクエストの本文で JSON ドキュメントを受信する場合で、トピックに対して発行された元の件名およびメッセージとまったく同じ表現が必要なときは、エスケープした文字を元の文字値に戻す変換を実行する必要があります。これは、署名では署名対象の文字列の一部としてメッセージと件名が元の形式で使用されるため、通知の署名を確認する場合に重要です。
3. Amazon SNS によって送信された通知、サブスクリプションの確認、またはサブスクリプション解除の確認メッセージの信頼性を確認する必要があります。エンドポイントは、Amazon SNS メッセージに含まれている情報を使用して署名を再作成し、署名を Amazon SNS がメッセージとともに送信した署名とマッチングすることで、メッセージの内容を確認できるようにします。メッセージの署名の確認の詳細については、「Amazon SNS メッセージの署名を検証する」を参照してください。
4. コードは、ヘッダーフィールド `x-amz-sns-message-type` で指定したタイプに基づいて、HTTP リクエストの本文に含まれている JSON ドキュメントを読み取り、メッセージを処理する必要があります。メッセージの 2 つの主要なタイプを処理するためのガイドラインを以下に示します。

   SubscriptionConfirmation

   `SubscribeURL` の値を読み取り、その URL にアクセスします。サブスクリプションを確認し、エンドポイントで通知の受信を開始するには、(URL に HTTP GET リクエストを送信するなどして) `SubscribeURL` URL にアクセスする必要があります。`SubscribeURL` の例については、前のステップの HTTP リクエストの例を参照してください。`SubscriptionConfirmation` メッセージの形式の詳細については、「HTTP/HTTPS サブスクリプションの確認の JSON 形式」を参照してください。URL にアクセスすると、以下の XML ドキュメントのような応答があります。ドキュメントは、`ConfirmSubscriptionResult` 要素内でエンドポイントのサブスクリプション ARN を返します。

```
<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>
```

SubscribeURL にアクセスする代わりに、ConfirmSubscription アクションを SubscriptionConfirmation メッセージの対応する値に設定した Token とともに使ってサブスクリプションを確認できます。トピックの所有者とサブスクリプションの所有者がエンドポイントのサブスクリプションを解除できるようにするには、AWS 署名とともに ConfirmSubscription アクションを呼び出します。

通知

Subject および Message の値を読み取り、トピックに発行された通知情報を取得します。

Notification メッセージの形式の詳細については、「HTTP/HTTPS ヘッダー」を参照してください。以下の HTTP POST リクエストは、エンドポイント example.com に送信される通知メッセージの例です。

```
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. エンドポイントが Amazon SNS からの HTTP POST メッセージに適切なステータスコードで応答することを確認します。接続は 15 秒でタイムアウトします。接続がタイムアウトする前にエンドポイントが応答しない場合、またはエンドポイントが 200～4xx の範囲外のステータスコードを返す場合、Amazon SNS はメッセージの配信に失敗したと見なします。
6. コードが Amazon SNS からのメッセージ配信の再試行を処理できることを確認します。Amazon SNS は、エンドポイントから正常な応答を受け取らない場合、そのメッセージの配信をもう一度試みます。これはサブスクリプションの確認メッセージを含むすべてのメッセージに適用されます。デフォルトでは、メッセージの最初の配信が失敗した場合、Amazon SNS は試行の失敗の間隔として 20 秒に設定された遅延時間で、最大 3 回再試行を試みます。

   > **Note**
   >
   > メッセージのリクエストは 15 秒後にタイムアウトします。つまり、メッセージ配信の失敗がタイムアウトによって起きた場合、Amazon SNS は前回の配信の試行から約 35 秒後に再試行します。エンドポイントに別の配信ポリシーを設定できます。

   Amazon SNS は `x-amz-sns-message-id` ヘッダーフィールドを使用して、Amazon SNS トピックにパブリッシュされた各メッセージを一意に識別します。受信メッセージとともに処理したメッセージ ID を比較することで、メッセージが再試行であるかどうかを判断することができます。
7. HTTPS エンドポイントをサブスクライブする場合、エンドポイントに、信頼された認定権限 (CA) からのサーバー証明書があることを確認します。Amazon SNS は、Amazon SNS によって信頼された CA によって署名されたサーバー証明書がある HTTPS エンドポイントにのみメッセージを送信します。
8. Amazon SNS メッセージを受信するために作成したコードをデプロイします。エンドポイントをサブスクライブするときに、エンドポイントは少なくともサブスクリプションの確認メッセージを受信する準備ができている必要があります。

ステップ 2: Amazon SNS トピックに HTTP/HTTPS エンドポイントをサブスクライブする

トピックを通じて HTTP または HTTPS エンドポイントにメッセージを送信するには、エンドポイントを Amazon SNS トピックにサブスクライブする必要があります。エンドポイントを指定するには、その URL を使用します。トピックにサブスクライブするには、Amazon SNS コンソール、sns-subscribe コマンド、または Subscribe API アクションを使用できます。開始する前に、サブスクライブするエンドポイントの URL があり、ステップ 1 で説明したように、エンドポイントで確認メッセージや通知メッセージを受信する準備ができていることを確認します。

Amazon SNS コンソールを使って HTTP または HTTPS エンドポイントをトピックにサブスクライブするには

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. [Create subscription] (サブスクリプションの作成) を選択します。
4. [Protocol] ドロップダウンリストで [HTTP] または [HTTPS] を選択します。
5. [エンドポイント] ボックスで、メッセージの送信先となるトピックのエンドポイントの URL をペーストし、[サブスクリプションの作成] を選択します。
6. 確認メッセージが表示されます。[Close] (閉じる) を選択します。

   新しいサブスクライブリプションの [サブスクリプション ID] に、PendingConfirmation と表示されます。サブスクリプションを確認すると、[サブスクリプション ID] にサブスクリプション ID が表示されます。

ステップ 3: サブスクリプションを確認する

エンドポイントをサブスクライブすると、Amazon SNS はエンドポイントにサブスクリプションの確認メッセージを送信します。ステップ 1 で説明しているアクションを実行するコードは、すでにエンドポイントにデプロイしています。特に、エンドポイントのコードは、サブスクリプションの確認メッセージから SubscribeURL 値を取得し、SubscribeURL そのもので指定された場所にアクセスするか、この場所を利用可能にして、手動で SubscribeURL にアクセスできるようにする必要があります (例: ウェブブラウザを使用している場合)。Amazon SNS は、サブスクリプションが確認されるまでエンドポイントにメッセージを送信しません。SubscribeURL にアクセスすると、応答にはサブスクリプションの ARN を指定する SubscriptionArn 要素を含む XML ドキュメントが含まれます。Amazon SNS コンソールを使用して、サブスクリプションが確認されたことを検証

できます。[サブスクリプション ID] には、最初にサブスクリプションを追加したときに表示された `PendingConfirmation` 値の代わりに、サブスクリプションの ARN が表示されます。

ステップ 4: サブスクリプションの配信ポリシーを設定する (オプション)

デフォルトでは、メッセージの最初の配信が失敗した場合、Amazon SNS は試行の失敗の間隔として 20 秒に設定された遅延時間で、最大 3 回再試行を試みます。ステップ 1 で説明しているように、エンドポイントには再試行されたメッセージを処理できるコードが必要です。トピックまたはサブスクリプションで配信ポリシーを設定することで、Amazon SNS が失敗したメッセージを再試行する頻度と間隔を制御できます。`DeliveryPolicy` で HTTP/S 通知のコンテンツタイプを指定することもできます。詳細については、「HTTP/S 配信ポリシーの作成」を参照してください。

ステップ 5: トピックに発行するアクセス権限をユーザーに付与する (オプション)

デフォルトでは、トピックの所有者にはトピックを発行するアクセス権限があります。その他のユーザーやアプリケーションがトピックに発行できるようにするには、AWS Identity and Access Management (IAM) を使ってトピックへの発行許可を付与する必要があります。Amazon SNS アクションに対するアクセス権限を IAM ユーザーに付与する方法の詳細については、「Amazon SNS でのアイデンティティベースのポリシーを使用する」を参照してください。

トピックへのアクセスは、以下の 2 つの方法でコントロールできます。

- IAM ユーザーまたはグループにポリシーを追加する。トピックへのアクセス権限をユーザーに付与する最も簡単な方法では、グループを作成し、適切なポリシーをグループに追加して、そのグループにユーザーを追加します。個々のユーザーに設定するポリシーを継続的に追跡するよりも、グループに対してユーザーを追加または削除する方がはるかに簡単です。
- トピックにポリシーを追加します。別の AWS アカウントにトピックへのアクセス権限を追加する場合、そのプリンシパルとして、アクセス権限を付与する AWS アカウントを持っているポリシーを追加することが唯一の方法です。

ほとんどの場合は、最初の方法 (ポリシーをグループに適用し、適切なユーザーをグループに追加または削除することでアクセス権限を管理する) を使用します。別のアカウントのユーザーにアクセス権限を付与する必要がある場合は、2 番目の方法を使用します。

IAM ユーザーまたはグループに次のポリシーを追加した場合、そのユーザーまたはそのグループのメンバーに、MyTopic トピックで `sns:Publish` アクションを実行する許可が付与されます。

```
{
```

```
  "Statement":[{
    "Sid":"AllowPublishToMyTopic",
    "Effect":"Allow",
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

以下の例のポリシーでは、トピックへのアクセス権限を別のアカウントに付与する方法を示しています。

> **Note**
>
> アカウントのリソースへの別の AWS アカウント アクセス権限を付与する場合、管理者レベルのアクセス (ワイルドカードアクセス) の権限を持っている IAM ユーザーにも、そのリソースへのアクセス許可が付与されます。他のアカウントの他のすべての IAM ユーザーは、自動的にリソースへのアクセスが拒否されます。その AWS アカウント アクセスの特定の IAM ユーザーにリソースへのアクセス権を付与する場合、管理者レベルアクセス権を持っているアカウントまたは IAM ユーザーは、そのリソースのアクセス権限をそれらの IAM ユーザーに委任する必要があります。クロスアカウントの委任の詳細については、『IAM ガイドの使用』の「Enabling Cross-Account Access」を参照してください。

アカウント 123456789012 の「マイトピック」トピックに以下のポリシーを追加した場合、そのトピックで sns:Publish アクションを実行するアクセス権限をアカウント 111122223333 に付与したことになります。

```
{
  "Statement":[{
    "Sid":"Allow-publish-to-topic",
    "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

ステップ 6: HTTP/HTTPS エンドポイントにメッセージを送信する

トピックに発行することで、トピックのサブスクリプションにメッセージを送信できます。トピックに発行するには、Amazon SNS コンソール、`sns-publish` CLI コマンド、または `Publish` API アクションを使用できます。

ステップ 1 を実行した場合、エンドポイントでデプロイしたコードで通知を処理できます。

Amazon SNS コンソールを使用してトピックに発行するには

1. トピックに発行するアクセス権限を持っている AWS アカウント アカウントまたは IAM ユーザーの認証情報を使用して、AWS Management Console にサインインし、Amazon SNS コンソール (https://console.aws.amazon.com/sns/) を開きます。
2. ナビゲーションパネルで、[トピック] を選択し、トピックを選択します。
3. [メッセージの発行] ボタンをクリックします。
4. [件名] ボックスに件名 (「**`Testing publish to my endpoint`**」など) を入力します。
5. [メッセージ] ボックスにテキスト (「**`Hello world!`**」など) を入力し、[メッセージの発行] を選択します。

   「Your message has been successfully published.」というメッセージが表示されます。

Amazon SNS メッセージの署名を検証する

Amazon SNS で HTTP エンドポイントに送信されたメッセージの信頼性を検証するには、メッセージの署名を検証できます。メッセージの信頼性の検証を推奨するケースは 2 つあります。まず、Amazon SNS が HTTP エンドポイントに対して、トピックにサブスクライブしたというメッセージを送信するときです。2 つ目は、`Subscribe` または `Unsubscribe` API アクションの実行時に Amazon SNS が HTTP エンドポイントに確認メッセージを送信する場合です。

Amazon SNS によって送信されたメッセージを検証するときに、以下のオペレーションを行う必要があります。

- Amazon SNS から証明書を取得するときは必ず HTTPS を使用する
- 証明書の信頼性を検証する
- 証明書が Amazon SNS から受信されたことを確認する。
- 可能であれば、サポートされている Amazon SNS 用の AWS SDK のいずれかを使用してメッセージを検証および確認する。

- Amazon SNS メッセージが希望する TopicArn から受信されていることを検証します。

Amazon SNS は次の 2 つのメッセージ署名バージョンをサポートしています。

- SignatureVersion1: Amazon SNS は、メッセージの SHA1 ハッシュに基づいて署名を作成します。
- SignatureVersion2: Amazon SNS は、メッセージの SHA256 ハッシュに基づいて署名を作成します。

Amazon SNS トピックでメッセージ署名バージョンを設定するには

Amazon SNS トピックは、デフォルトで SignatureVersion 1 を使用します。Amazon SNS トピックのハッシュアルゴリズム、SignatureVersion 1 (SHA1) または SignatureVersion2 (SHA256) を選択するには、SetTopicAttributes API アクションを使用できます。

次のコード例は、AWS CLI を使用してトピック属性 SignatureVersion を設定する方法を示しています。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

HTTP クエリベースのリクエストの使用時に Amazon SNS メッセージの署名を確認するには

1. Amazon SNS がエンドポイントに送信した HTTP POST リクエストの本文で、JSON ドキュメントから名前と値のペアを抽出します。署名対象の文字列を作成するために、名前と値のペアの一部の値を使用します。Amazon SNS メッセージの署名を確認する場合は、Message および Subject の値で、エスケープした制御文字を元の文字表現に変換し直すことが重要です。これらの値では、署名対象の文字列の一部として使用するときに、元の形式になっている必要があります。JSON ドキュメントを解析する方法の詳細については、「ステップ 1: エンドポイントで Amazon SNS メッセージを処理する準備が完了していることを確認する」を参照してください。

   SignatureVersion は、Amazon SNS がメッセージの署名を生成するために使用する署名バージョンを示します。署名バージョンからは、署名を生成する方法の要件を判断できます。通知では、Amazon SNS は現在、署名バージョン 1 および 2 をサポートしています。このセクションでは、これらの署名バージョンを使用して署名を作成する手順を説明します。

2. メッセージの署名に Amazon SNS が使用した X509 証明書を取得します。SigningCertURL 値は、メッセージのデジタル署名の作成に使用された X509 証明書の場所を指します。この場所から証明書を取得します。
3. 証明書から公開キーを抽出します。SigningCertURL で指定された証明書からの公開キーを使用して、メッセージの信頼性と整合性を確認します。
4. メッセージタイプを判断します。署名対象の文字列の形式は Type 値によって指定されるメッセージタイプによって異なります。
5. 署名対象の文字列を作成します。署名対象の文字列は、改行文字で区切られた、メッセージからの特定の名前と値のペアのリストです。それぞれの名前と値のペアは、名前で始まり、改行文字、値の順に続き、改行文字で終わります。名前と値のペアは、バイト順でソートしてリストされる必要があります。

   署名対象の文字列には、メッセージタイプに応じて以下の名前と値のペアが必要です。

   通知

   通知メッセージには、以下の名前と値のペアを含める必要があります。

```
Message
MessageId
Subject (if included in the message)
Timestamp
TopicArn
Type
```

   以下の例は Notification の署名対象の文字列です。

```
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 および UnsubscribeConfirmation

SubscriptionConfirmation および UnsubscribeConfirmation メッセージには、以下の名前と値のペアを含める必要があります。

```
Message
MessageId
SubscribeURL
Timestamp
Token
TopicArn
Type
```

以下の例は SubscriptionConfirmation の署名対象の文字列です。

```
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
```

6. Base64 形式から Signature 値をデコードします。メッセージは、Base64 としてエンコードされた Signature 値で署名を配信します。署名の値を、計算した署名と比較する前に、Base64 から Signature の値をデコードし、同じ形式を使って値を比較します。
7. Amazon SNS メッセージの派生したハッシュ値を生成します。署名の生成に使用されるのと同じハッシュアルゴリズムに、正規形式で Amazon SNS メッセージを送信します。
   a. SignatureVersion が 1 のとき SHA1 をハッシュアルゴリズムとして使用します。
   b. SignatureVersion が 2 のとき SHA256 をハッシュアルゴリズムとして使用します。

8. Amazon SNS メッセージのアサートされたハッシュ値を生成します。アサートされたハッシュ値は、Amazon SNS メッセージで配信された署名を復号するために (ステップ 3 の) 公開キーバリューを使用した結果です。
9. Amazon SNS メッセージの信頼性と整合性を確認します。(ステップ 7 から) 派生したハッシュ値を、(ステップ 8 から) アサートされたハッシュ値に比較します。値が同じ場合、受信者はメッセージが転送中に変更されておらず、メッセージが Amazon SNS から発信されたことを確認できます。値が同じでない場合、受信者はこれを信頼することはできません。

メッセージ形式を解析する

Amazon SNS では以下の形式が使用されます。

トピック

- HTTP/HTTPS ヘッダー
- HTTP/HTTPS サブスクリプションの確認の JSON 形式
- HTTP/HTTPS 通知の JSON 形式
- HTTP/HTTPS サブスクリプションの解除の JSON 形式
- SetSubscriptionAttributes 配信ポリシーの JSON 形式
- SetTopicAttributes 配信ポリシーの JSON 形式

HTTP/HTTPS ヘッダー

Amazon SNS がサブスクリプションの確認、通知、またはサブスクリプション解除の確認メッセージを HTTP/HTTPS エンドポイントに送信するときは、Amazon SNS 固有の多くのヘッダー値とともに POST メッセージを送信します。ヘッダー値を使用すると、JSON メッセージ本文を解析して Type 値を読み取ることなく、メッセージタイプの識別などのタスクを実行できます。デフォルトでは、Amazon SNS は、`Content-Type` が `text/plain; charset=UTF-8` に設定されている HTTP/S エンドポイントにすべての通知を送信します。text/plain (デフォルト) 以外の `Content-Type` を選択するには、「HTTP/S 配信ポリシーの作成」の「`headerContentType`」を参照してください。

`x-amz-sns-message-type`

メッセージのタイプ。指定できる値は、`SubscriptionConfirmation`、`Notification` および `UnsubscribeConfirmation` です。

x-amz-sns-message-id

発行するメッセージごとの汎用一意識別子 (UUID)。再試行間に Amazon SNS が再送信する通知の場合、元のメッセージのメッセージ ID が使用されます。

x-amz-sns-topic-arn

このメッセージの発行先トピックの Amazon リソースネーム (ARN)。

x-amz-sns-subscription-arn

このエンドポイントへのサブスクリプションの ARN。

次の HTTP POST ヘッダーは、HTTP エンドポイントへの `Notification` メッセージのヘッダーの例です。

```
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 サブスクリプションの確認の JSON 形式

HTTP/HTTPS エンドポイントをサブスクライブすると、Amazon SNS は HTTP/HTTPS エンドポイントにサブスクリプションの確認メッセージを送信します。このメッセージには、サブスクリプションを確認するためにアクセスする必要がある `SubscribeURL` 値が含まれています (または、`Token` 値を `ConfirmSubscription` で使用できます)。

> **Note**
>
> サブスクリプションが確認されるまで、Amazon SNS はこのエンドポイントに通知を送信しません。

サブスクリプションの確認メッセージは、以下の名前と値のペアを持つ JSON ドキュメントを含むメッセージ本文がある POST メッセージです。

Type

メッセージのタイプ。サブスクリプションの確認の場合、タイプは SubscriptionConfirmation です。

MessageId

発行するメッセージごとの汎用一意識別子 (UUID)。再試行間に Amazon SNS が再送信するメッセージの場合、元のメッセージのメッセージ ID が使用されます。

Token

サブスクリプションを確認するために ConfirmSubscription アクションで使用できる値。または、SubscribeURL にアクセスしても構いません。

TopicArn

このエンドポイントがサブスクライブしているトピックの Amazon リソースネーム (ARN)。

Message

メッセージについて説明する文字列。サブスクリプションの確認の場合、この文字列は以下のようになります。

```
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

サブスクリプションを確認するためにアクセスする必要がある URL。または、代わりに Token を ConfirmSubscription アクションで使用してサブスクリプションを確認することもできます。

Timestamp

サブスクリプションの確認が送信された日時 (GMT)

SignatureVersion

使用される Amazon SNS 署名のバージョン

- SignatureVersion が 1 である場合、Signature は Message、MessageId、Type、Timestamp、TopicArn の各値の Base64 でエンコードされた SHA1withRSA シグネチャです。
- SignatureVersion が 2 である場合、Signature は Message、MessageId、Type、Timestamp、TopicArn の各値の Base64 でエンコードされた SHA256withRSA シグネチャです。

Signature

Message、MessageId、Type、Timestamp、TopicArn の各値の BaseBase64 でエンコードされた SHA1withRSA または SHA256withRSA シグネチャ。

SigningCertURL

メッセージの署名に使用された証明書の URL

次の HTTP POST メッセージは、HTTP エンドポイントへの SubscriptionConfirmation メッセージの例です。

```
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 通知の JSON 形式

Amazon SNS がサブスクライブしている HTTP または HTTPS エンドポイントに通知を送信すると、エンドポイントに送信される POST メッセージには、以下の名前と値のペアを持つ JSON ドキュメントを含むメッセージ本文があります。

`Type`

メッセージのタイプ。通知の場合、タイプは `Notification` です。

`MessageId`

発行するメッセージごとの汎用一意識別子 (UUID)。再試行間に Amazon SNS が再送信する通知の場合、元のメッセージのメッセージ ID が使用されます。

`TopicArn`

このメッセージの発行先トピックの Amazon リソースネーム (ARN)。

`Subject`

通知をトピックに発行したときに指定した `Subject` パラメータ。

> **Note**
>
> このパラメータはオプションです。`Subject` を指定しなかった場合、この名前と値のペアは、この JSON ドキュメントに表示されません。

`Message`

通知をトピックに発行したときに指定した `Message` 値。

`Timestamp`

通知が公開されたときの日時 (GMT)。

`SignatureVersion`

使用される Amazon SNS 署名のバージョン

- SignatureVersion が 1 である場合、Signature は Message、MessageId、Subject (存在する場合)、Type、Timestamp、TopicArn、TopicArn の各値の Base64 でエンコードされた SHA1withRSA シグネチャです。
- SignatureVersion が 2 である場合、Signature は Message、MessageId、Subject (存在する場合)、Type、Timestamp、TopicArn の各値の Base64 でエンコードされた SHA256withRSA シグネチャです。

Signature

Message、MessageId、Subject (存在する場合)、Type、Timestamp、TopicArn の各値の Base64 でエンコードされた SHA1withRSA または SHA256withRSA シグネチャ。

SigningCertURL

メッセージの署名に使用された証明書の URL

UnsubscribeURL

このトピックからエンドポイントのサブスクリプションを解除するために使用できる URL。この URL にアクセスすると、Amazon SNS はエンドポイントのサブスクリプションを解除し、このエンドポイントへの通知の送信を停止します。

次の HTTP POST メッセージは、HTTP エンドポイントへの Notification メッセージの例です。

```
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 サブスクリプションの解除の JSON 形式

HTTP/HTTPS エンドポイントのサブスクリプションがトピックから解除されると、Amazon SNS はエンドポイントにサブスクリプションの確認メッセージを送信します。

サブスクリプション解除の確認メッセージは、次の名前と値のペアを持つ JSON ドキュメントを含むメッセージ本文がある POST メッセージです。

`Type`

メッセージのタイプ。サブスクリプション解除の確認の場合、タイプは `UnsubscribeConfirmation` です。

`MessageId`

発行するメッセージごとの汎用一意識別子 (UUID)。再試行間に Amazon SNS が再送信するメッセージの場合、元のメッセージのメッセージ ID が使用されます。

`Token`

サブスクリプションを再確認するために [ConfirmSubscription](#) アクションで使用できる値。または、`SubscribeURL` にアクセスしても構いません。

`TopicArn`

このエンドポイントがサブスクリプションを解除されたトピックの Amazon リソースネーム (ARN)。

`Message`

メッセージについて説明する文字列。サブスクリプション解除の確認の場合、この文字列は次のようになります。

```
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

サブスクリプションを再確認するためにアクセスする必要がある URL。または、代わりに Token を ConfirmSubscription アクションで使用してサブスクリプションを再確認することもできます。

Timestamp

サブスクリプション解除の確認が送信された日時 (GMT)。

SignatureVersion

使用される Amazon SNS 署名のバージョン

- SignatureVersion が 1 である場合、Signature は Message、MessageId、Type、Timestamp、TopicArn の各値の Base64 でエンコードされた SHA1withRSA シグネチャです。
- SignatureVersion が 2 である場合、Signature は Message、MessageId、Type、Timestamp、TopicArn の各値の Base64 でエンコードされた SHA256withRSA シグネチャです。

Signature

Message、MessageId、Type、Timestamp、TopicArn の各値の BaseBase64 でエンコードされた SHA1withRSA または SHA256withRSA シグネチャ。

SigningCertURL

メッセージの署名に使用された証明書の URL

次の HTTP POST メッセージは、HTTP エンドポイントへの UnsubscribeConfirmation メッセージの例です。

```
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 配信ポリシーの JSON 形式

SetSubscriptionAttributes アクションにリクエストを送信し、AttributeName パラメータを DeliveryPolicy の値に設定する場合、AttributeValue パラメータの値は有効な JSON のオブジェクトである必要があります。例えば、次の例では、配信ポリシーを合計 5 回の再試行に設定します。

```
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}}
...
```

AttributeValue パラメータの値には、次の JSON 形式を使用します。

```
{
    "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"
    }
}
```

SetSubscriptionAttribute アクションの詳細については、「Amazon Simple Notification Service API リファレンス」の「SetSubscriptionAttributes」を参照してください。サポートされている HTTP コンテンツタイプヘッダーの詳細については、「HTTP/S 配信ポリシーの作成」を参照してください。

SetTopicAttributes 配信ポリシーの JSON 形式

SetTopicAttributes アクションにリクエストを送信し、AttributeName パラメータを DeliveryPolicy の値に設定する場合、AttributeValue パラメータの値は有効な JSON オブジェクトである必要があります。例えば、次の例では、配信ポリシーを合計 5 回の再試行に設定します。

```
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}}}
...
```

AttributeValue パラメータの値には、次の JSON 形式を使用します。

```
{
```

```
    "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"
        }
    }
}
```

SetTopicAttribute アクションの詳細については、「Amazon Simple Notification Service API リファレンス」のSetTopicAttributesにアクセスしてください。サポートされている HTTP コンテンツタイプヘッダーの詳細については、「HTTP/S 配信ポリシーの作成」を参照してください。

AWS Event Fork Pipelines へのファンアウト

イベントのアーカイブと分析のために、Amazon SNS は Amazon Data Firehose とのネイティブ統合の使用を推奨するようになりました。Firehose 配信ストリームを SNS トピックにサブスクライブできます。これにより、Amazon Simple Storage Service (Amazon S3) バケット、Amazon Redshift テーブル、Amazon OpenSearch Service (OpenSearch サービス) などのアーカイブエンドポイントと分析エンドポイントに通知を送信できます。Firehose 配信ストリームで Amazon SNS を使用することは、 AWS Lambda関数を使用する必要のない、フルマネージド型のコードレスソリューションです。詳細については、「Firehose 配信ストリームへのファンアウト」を参照してください。

Amazon SNS で構築したイベント駆動型アプリケーションで受信者サービスを使用し、発行者サービスでトリガーされたイベントに応答して自動的に作業を実行できます。このアーキテクチャパターンにより、サービスの再利用性、相互運用性、およびスケーラビリティを高めることができます。ただし、イベントのストレージ、バックアップ、検索、分析、再生などの一般的なイベント処理要件に対応するパイプラインにイベント処理を分岐させることは多大な労力を要する場合があります。

イベント駆動型アプリケーションの開発を迅速化するために、AWS Event Fork Pipelines による、イベント処理パイプラインを Amazon SNS トピックへサブスクライブできます。AWSEvent Fork Pipelines は、AWS サーバーレスアプリケーションモデル (AWS SAM) に基づいた、オープンソースのネストされたアプリケーションのスイートであり、AWS Event Fork Pipelines スイート ([Show apps that create custom IAM roles or resource policies] (カスタム IAM ロールまたはリソースポリシーを作成するアプリケーションの表示) を選択) から AWS アカウントへ直接デプロイできます。

AWS Event Fork Pipelines のユースケースについては、「AWS Event Fork Pipelines サンプルアプリケーションをデプロイしてテストする」を参照してください。

トピック

- AWS Event Fork Pipelines の仕組み
- AWS Event Fork Pipelines のデプロイ
- AWS Event Fork Pipelines サンプルアプリケーションをデプロイしてテストする
- Amazon SNS トピックに AWS Event Fork Pipelines をサブスクライブする。

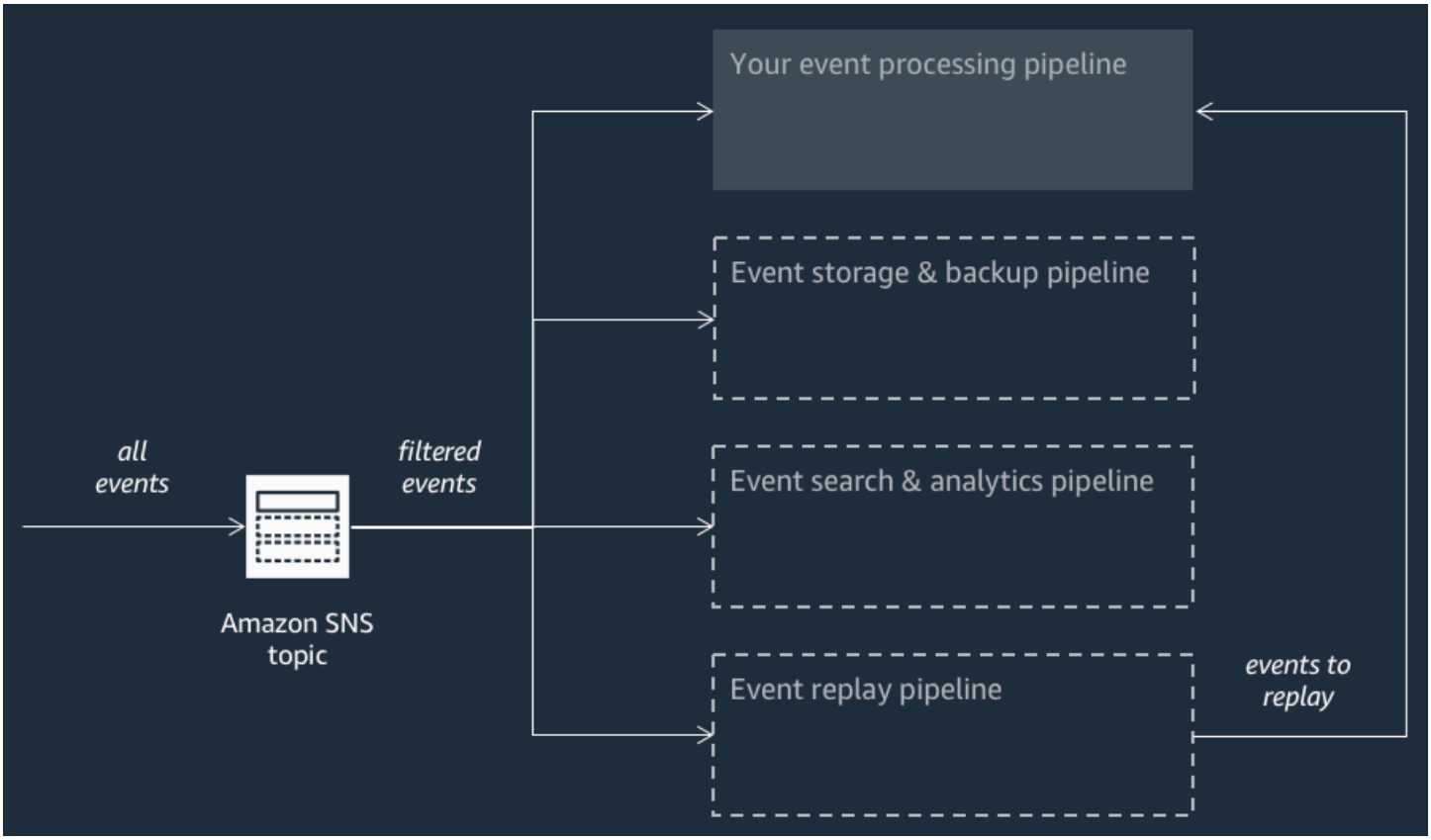
AWS Event Fork Pipelines の仕組み

AWS Event Fork Pipelines は、サーバーレスな設計パターンです。ただし、AWS SAM に基づいてネストされたサーバーレスアプリケーションのスイートでもあります (これを AWS Serverless Application Repository (AWS SAR) から AWS アカウント に直接デプロイしてイベント駆動型のプラットフォームを強化できます)。アーキテクチャの必要に応じて、これらのネストされたアプリケーションを個別にデプロイできます。

トピック

- イベントのストレージおよびバックアップパイプライン
- イベントの検索および分析パイプライン
- イベントの再生パイプライン

次の図は、3 つのネストされたアプリケーションで補完された AWS Event Fork Pipelines アプリケーションを示しています。アーキテクチャの必要に応じて、AWS SAR の AWS Event Fork Pipelines スイートから任意のパイプラインを個別にデプロイできます。



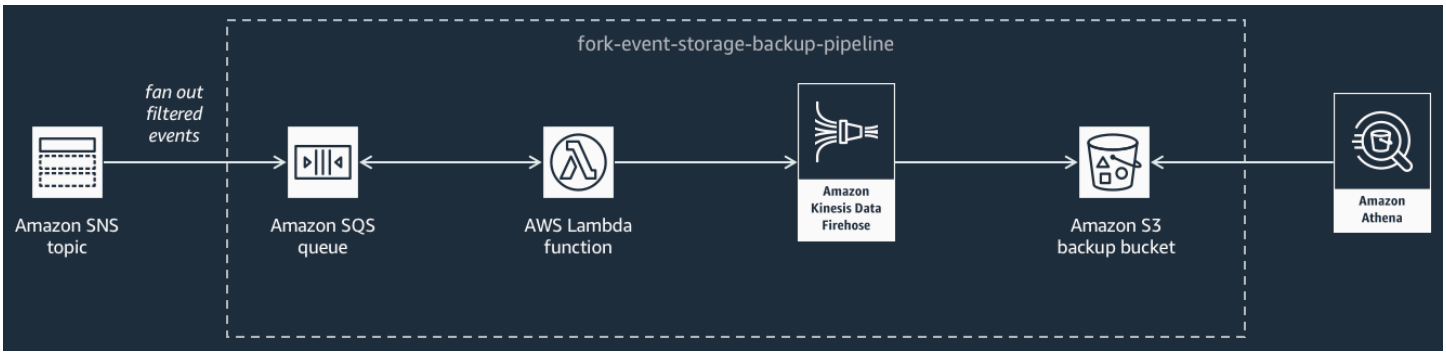
各パイプラインは、同じ Amazon SNS トピックにサブスクライブされるため、このトピックに発行された複数のイベントを並列処理できます。各パイプラインは独立しており、独自のサブスクリプションフィルターポリシーを設定できます。これにより、パイプラインでは、トピックに発行されたすべてのイベントではなく、関心のあるイベントのサブセットに絞って処理できます。

> **Note**
>
> 通常のイベント処理パイプラインに加えて 3 つの AWS Event Fork Pipelines (おそらくは Amazon SNS トピックにサブスクライブ済み) を配置しているため、既存のワークロードで AWS Event Fork Pipelines を利用するために現在のメッセージ発行者の一部を変更する必要はありません。

イベントのストレージおよびバックアップパイプライン

次の図は、イベントのストレージおよびバックアップパイプラインを示しています。このパイプラインを Amazon SNS トピックにサブスクライブして、システムを通過するイベントを自動的にバックアップできます。

このパイプラインは、Amazon SQSAmazon SNSキュー、キュー内のこれらのイベントを自動的にポーリングして Amazon Data Firehose ストリームにプッシュする AWS Lambda関数、およびストリームによってロードされたイベントを永続的にバックアップする Amazon S3 バケットで構成されます。

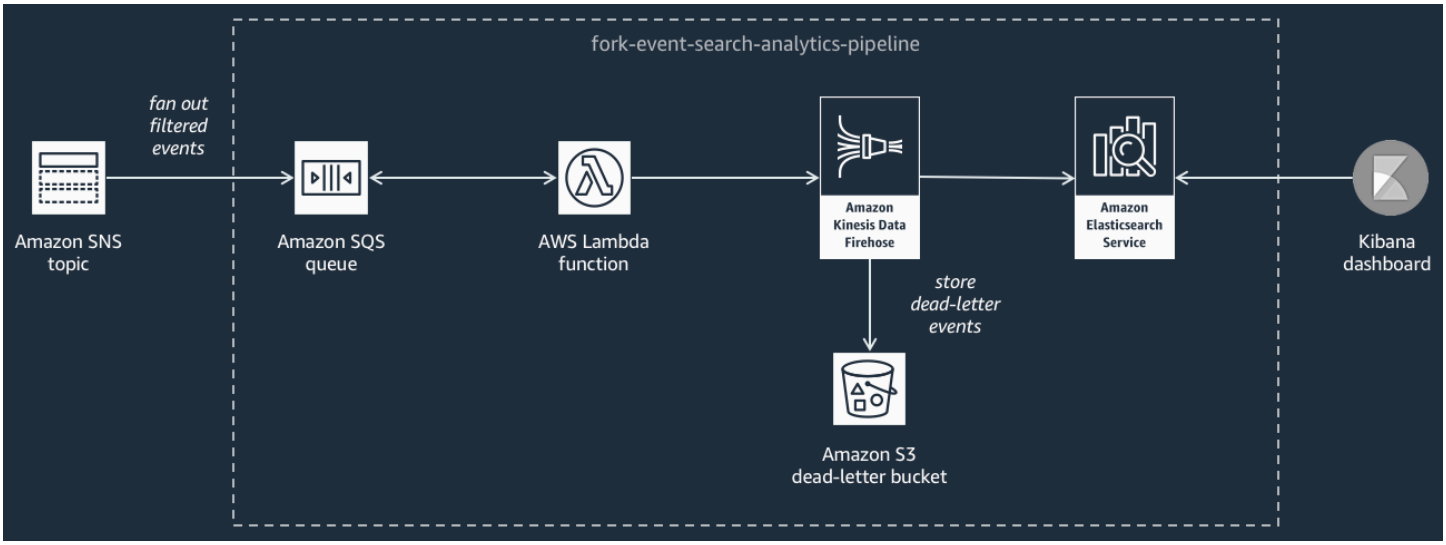


Firehose ストリームの動作を微調整するには、イベントをバケット内にロードする前に、イベントをバッファ処理、変換、および圧縮するようにストリームを設定できます。イベントがロードされたら、Amazon Athena で標準の SQL クエリを使用し、バケットに対してクエリを実行できます。既存の Amazon S3 バケットを再利用したり、新しいバケットを作成したりするようにパイプラインを設定することもできます。

イベントの検索および分析パイプライン

次の図は、イベントの検索および分析パイプラインを示しています。このパイプラインを Amazon SNS トピックにサブスクライブして、システムを通過するイベントを検索ドメインでインデックス付けし、これらのイベントに対して分析を実行できます。

このパイプラインは、Amazon SQSAmazon SNS キュー、キューからイベントをポーリングして Amazon Data Firehose ストリームにプッシュする AWS Lambda関数、Firehose ストリームによってロードされたイベントのインデックスを作成する Amazon OpenSearch サービスドメイン、および検索ドメインでインデックス作成できないデッドレターイベントを保存する Amazon S3 バケットで構成されます。



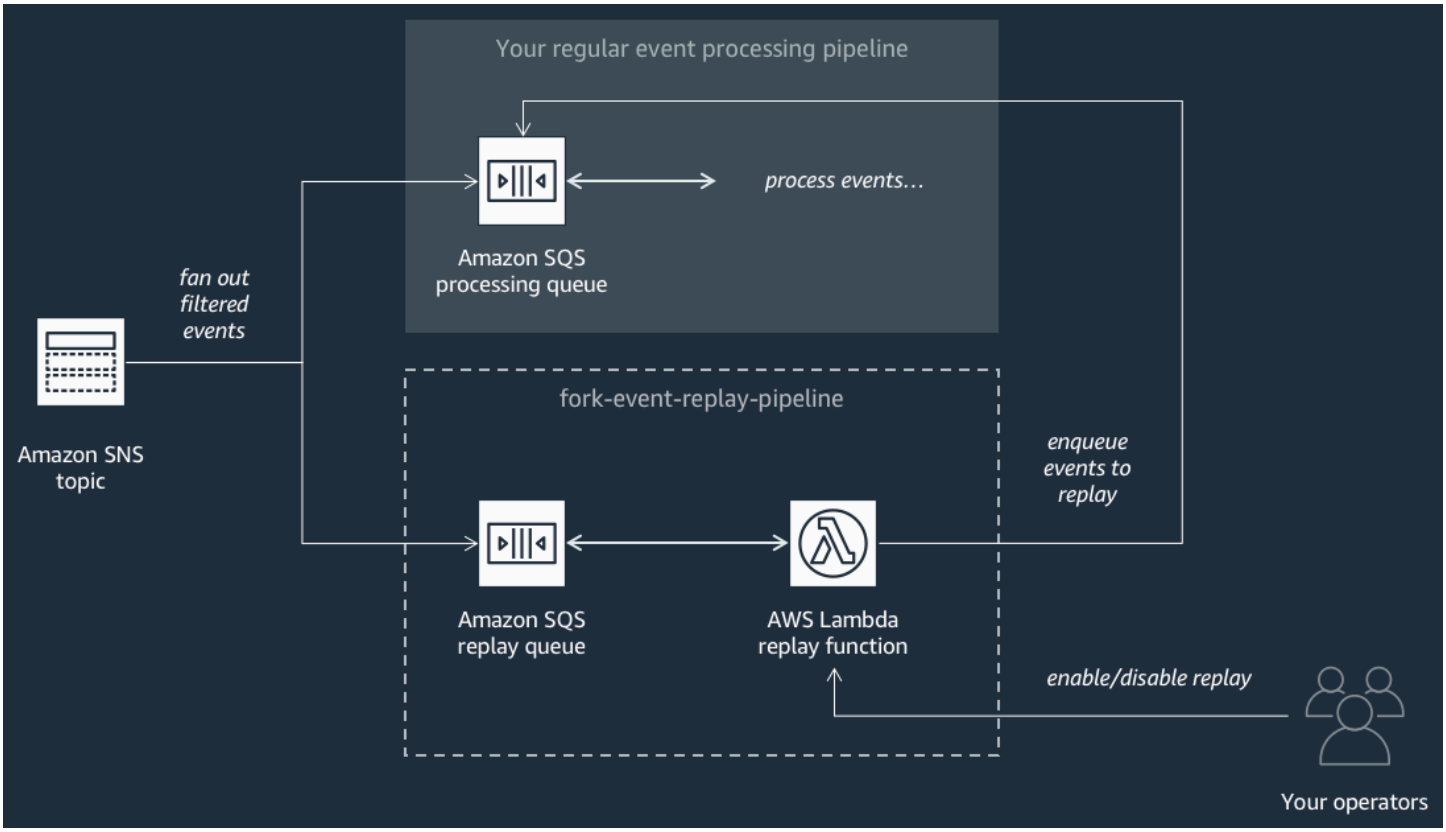
Firehose ストリームについてイベントのバッファ処理、変換、および圧縮を微調整するには、このパイプラインを設定できます。

パイプラインで 内の既存の OpenSearch ドメインを再利用するか、新しいドメインAWS アカウントを作成するかを設定することもできます。イベントは検索ドメインでインデックス付けされるため、Kibana を使用してイベントに対して分析を実行し、リアルタイムでビジュアルダッシュボードを更新できます。

イベントの再生パイプライン

次の図は、イベントの再生パイプラインを示しています。過去 14 日間にシステムで処理されたイベントを記録するには (プラットフォームを障害から復旧する必要がある場合など)、このパイプラインを Amazon SNS トピックにサブスクライブしてイベントを再処理できます。

このパイプラインは、Amazon SNS トピックから配信されたイベントをバッファ処理する Amazon SQS キューと、キューのイベントをポーリングして通常のイベント処理パイプライン (トピックにサブスクライブ済み) 内に再ルーティングする AWS Lambda 関数で構成されます。



> **Note**
>
> デフォルトでは、再生関数は無効になっており、イベントを再ルーティングしません。イベントを再処理する必要がある場合は、Amazon SQS 再生関数のイベントソースとして AWS Lambda 再生キューを有効にする必要があります。

AWS Event Fork Pipelines のデプロイ

AWS Event Fork Pipelines スイート ([カスタム IAM ロールまたはリソースポリシーを作成するアプリケーションを表示する] を選択) が AWS Serverless Application Repository でパブリックアプリケーションのグループとして使用可能であり、ここで AWS Lambda コンソールを使用して手動でスイートをデプロイしてテストできます。AWS Lambda コンソールを使用したパイプラインのデプロイについては、「Amazon SNS トピックに AWS Event Fork Pipelines をサブスクライブする。」を参照してください。

本稼働シナリオでは、AWS Event Fork Pipelines をアプリケーション全体の AWS SAM テンプレート内に埋め込むことをお勧めします。これをネストされたアプリケーション機能を使用して行うに

は、AWS SAM テンプレートにリソース AWS::Serverless::Application を追加し、ネストされたアプリケーションの AWS SAR ApplicationId と SemanticVersion を参照します。

例えば、AWS SAM テンプレートの Resources セクションに次の YAML スニペットを追加することで、イベントのストレージおよびバックアップパイプラインを、ネストされたアプリケーションとして使用できます。

```
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
```

パラメータ値を指定するときに、AWS CloudFormation 組み込み関数を使用してテンプレート内の他のリソースを参照できます。例えば、上の YAML スニペットで、TopicArn パラメータは、AWS SAM テンプレート内に定義されている AWS::SNS::Topic リソース MySNSTopic を参照します。詳細については、『AWS CloudFormation ユーザーガイド』の「組み込み関数リファレンス」を参照してください。

> **Note**
>
> AWS SAR アプリケーションの AWS Lambda コンソールページには、AWS SAR アプリケーションをネストするために必要な YAML をクリップボードにコピーする [SAM リソースとしてコピー] ボタンが含まれています。

AWS Event Fork Pipelines サンプルアプリケーションをデプロイしてテストする

イベント駆動型アプリケーションの開発を迅速化するために、AWS Event Fork Pipelines により、イベント処理パイプラインを Amazon SNS トピックへサブスクライブできます。AWSEvent Fork Pipelines は、AWS サーバーレスアプリケーションモデル (AWS SAM) に基づいた、オープンソースのネストされたアプリケーションのスイートであり、AWS Event Fork Pipelines スイート ([Show

apps that create custom IAM roles or resource policies] (カスタム IAM ロールまたはリソースポリシーを作成するアプリケーションの表示) を選択) から AWS アカウントへ直接デプロイできます。詳細については、「AWS Event Fork Pipelines の仕組み」を参照してください。

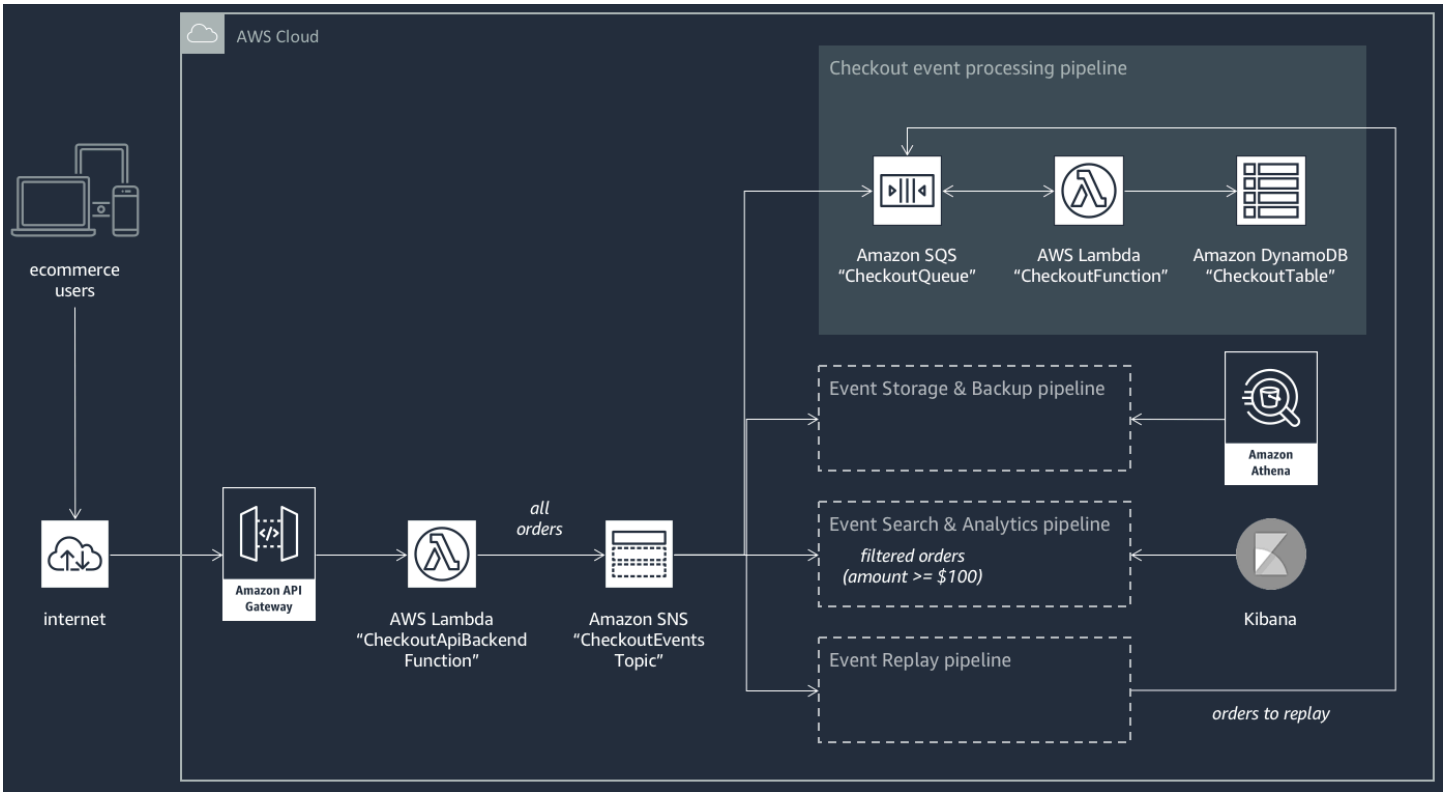
このページでは、AWS Management Console を使用して AWS Event Fork Pipelines サンプルアプリケーションをデプロイしてテストする方法を説明します。

> **Important**
>
> AWS Event Fork Pipelines サンプルアプリケーションのデプロイの完了後、不要な料金が発生しないように、AWS CloudFormation スタックは削除してください。詳細については、『AWS CloudFormation ユーザーガイド』の「AWS CloudFormation コンソールでスタックを削除する」を参照してください。

トピック

AWS Event Fork Pipelines のユースケースの例

次のシナリオでは、AWS Event Fork Pipelines を使用するイベント駆動型のサーバーレス e コマースアプリケーションについて説明します。この例の e コマースアプリケーションを で使用しAWS Serverless Application Repository、 AWS LambdaコンソールAWS アカウントを使用して にデプロイできます。ここでは、 でソースコードをテストして調査できます GitHub。



この e コマースアプリケーションでは、API Gateway でホストされ、AWS Lambda 関数 `CheckoutApiBackendFunction` で支援された RESTful API を通じて購入者から注文を受けます。この関数は、すべての受注を `CheckoutEventsTopic` という名前の Amazon SNS トピックに発行します。このトピックでは、これらの受注を 4 つの異なるパイプラインにファンアウトします。

最初のパイプラインは、e コマースアプリケーションの所有者によって設計および実装された通常のチェックアウト処理パイプラインです。このパイプラインには、すべての受注をバッファ処理する Amazon SQS `CheckoutQueue` キュー 、キューをポーリングしてこれらの注文を処理する、`CheckoutFunction` という名前の AWS Lambda 関数、およびすべての受注を安全に保存する DynamoDB テーブル `CheckoutTable` があります。

AWS Event Fork Pipelines を適用する

e コマースアプリケーションのコンポーネントは、主要ビジネスロジックを処理します。ただし、e コマースアプリケーションの所有者は以下にも対処する必要があります。

- コンプライアンス - 安全な、圧縮されたバックアップの保管時の暗号化と機密情報のサニタイズ
- レジリエンス - フルフィルメントプロセスの中断が発生した場合の最新の注文の再生
- 検索可能性 - 受注に対する分析の実行とメトリクスの生成

このイベント処理ロジックを実装する代わりに、アプリケーション所有者は AWS Event Fork Pipelines を `CheckoutEventsTopic` Amazon SNS トピックにサブスクライブできます。

- イベントのストレージおよびバックアップパイプライン は、データを変換して、クレジットカードの詳細の削除、データの 60 秒間のバッファ処理、GZIP を使用した圧縮、Amazon S3 のデフォルトのカスタマーマネージドキーを使用した暗号化を行うように設定されています。このキーは、AWS によって管理され、AWS Key Management Service (AWS KMS) により動作します。

  詳細については、「Amazon Data Firehose デベロッパーガイド」の「送信先として Amazon S3 を選択する」、「Amazon Data Firehose データ変換」、および「設定を構成する」を参照してください。
- イベントの検索および分析パイプライン には、30 秒間のインデックス再試行期間、検索ドメインでインデックスの作成に失敗した注文を保存するためのバケット、およびインデックスを作成した注文のセットを制限するためのフィルターポリシーが設定されています。

  詳細については、「Amazon Data Firehose デベロッパーガイド」の「送信先に OpenSearch サービスを選択する」を参照してください。
- イベントの再生パイプライン には、e コマースアプリケーションの所有者によって設計および実装された通常の注文処理パイプラインの Amazon SQS キュー部分が設定されています。

  詳細については、『Amazon Simple Queue Service デベロッパーガイド』の「キーの名前と URL」を参照してください。

次の JSON フィルターポリシーは、イベントの検索および分析パイプラインの設定で設定されます。これは、合計金額が 100 USD 以上の受注とのみ一致します。詳細については、「Amazon SNS メッセージフィルター処理」を参照してください。

```
{
   "amount": [{ "numeric": [ ">=", 100 ] }]
}
```

AWS Event Fork Pipelines パターンを使用すると、e コマースアプリケーションの所有者は、イベント処理用の差別化につながらないロジックのコーディングに伴いがちな開発オーバーヘッドを回避できます。代わりに、所有者は AWS Event Fork Pipelines を AWS Serverless Application Repository から AWS アカウント 内に直接デプロイできます。

ステップ 1: サンプルアプリケーションをデプロイするには

1. AWS Lambda コンソール にサインインします。
2. ナビゲーションパネルで [関数] を選択し、[関数の作成] を選択します。
3. [関数の作成] ページで、次の操作を実行します。
   a. [サーバーレスアプリケーションリポジトリの参照]、[パブリックアプリケーション]、[カスタム IAM ロールまたはリソースポリシーを作成するアプリケーションの表示] を選択します。
   b. `fork-example-ecommerce-checkout-api` を検索し、このアプリケーションを選択します。
4. fork-example-ecommerce-checkout-api ページで、次の操作を行います。
   a. [アプリケーション設定] セクションで、[アプリケーション名] に名前 (`fork-example-ecommerce-my-app` など) を入力します。

      > **Note**
      >
      > - リソースを後で簡単に見つけられるように、プレフィックス `fork-example-ecommerce` を保持します。
      > - 名前は、デプロイごとに一意にする必要があります。アプリケーション名を再利用すると、デプロイでは、スタックを新規作成せずに、以前にデプロイした AWS CloudFormation スタックを単に更新します。

   b. (オプション) アプリケーションの Lambda 関数を実行するには、次のいずれかLogLevelの設定を入力します。
      - `DEBUG`
      - `ERROR`
      - `INFO` (デフォルト)
      - `WARNING`
5. [このアプリケーションがカスタム IAM ロール、リソースポリシーを作成し、ネストされたアプリケーションをデプロイすることを承認します] を選択して、ページの下部にある [デプロイ] を選択します。

fork-example-ecommerce-*my-app* のデプロイステータス ページに、Lambda はアプリケーションをデプロイ中のステータスを表示します。

[リソース] セクションで、AWS CloudFormation によってスタックの作成が開始され、各リソースの [CREATE_IN_PROGRESS] ステータスが表示されます。プロセスが完了すると、AWS CloudFormationに [CREATE_COMPLETE]ステータスが表示されます。

> **Note**
>
> すべてのリソースがデプロイされるまで 20 ~ 30 分かかる場合があります。

デプロイが完了すると、Lambda によって [アプリケーションはデプロイ済みです] ステータスが表示されます。

ステップ 2: サンプルアプリケーションを実行するには

1. AWS Lambda コンソールのナビゲーションパネルで、[アプリケーション] を選択します。
2. [アプリケーション] ページの検索フィールドで、`serverlessrepo-fork-example-ecommerce-`*`my-app`* を検索し、そのアプリケーションを選択します。
3. [リソース] セクションで、以下の操作を行います。
   a. タイプが であるリソースを検索するにはApiGatewayRestApi、 などのタイプ でリソースをソートServerlessRestApiし、リソースを展開します。
   b. ApiGateway デプロイタイプとApiGatewayステージタイプの 2 つのネストされたリソースが表示されます。
   c. リンク [Prod API エンドポイント] をコピーし、これに `/checkout` を付加します。次に例を示します。

   ```
   https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
   ```

4. 次の JSON を `test_event.json` という名前のファイルにコピーします。

   ```
   {
      "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. API エンドポイントに HTTPS リクエストを送信するには、curl コマンドを実行してサンプルイベントペイロードを入力として渡します。次に例を示します。

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

API は、次の空のレスポンスを返し、実行が成功したことを示します。

```
{ }
```

ステップ 3: サンプルアプリケーションとそのパイプラインの実行を検証するには

ステップ 1: サンプルのチェックアウトパイプラインの実行を検証するには

1. Amazon DynamoDB コンソールにサインインします。
2. ナビゲーションパネルで、[テーブル] を選択します。
3. `serverlessrepo-fork-example` を検索して `CheckoutTable` を選択します。
4. テーブルの詳細ページで [項目] を選択し、作成済みの項目を選択します。

   保存済みの属性が表示されます。

ステップ 2: イベントのストレージおよびバックアップパイプラインの実行を検証するには

1. [Amazon S3 コンソール] にサインインします。
2. ナビゲーションパネルで [バケット] を選択します。
3. `serverlessrepo-fork-example` を検索して、`CheckoutBucket` を選択します。
4. `.gz` を拡張子とするファイルが見つかるまで、ディレクトリ階層を移動します。
5. ファイルをダウンロードするには、[アクション]、[開く] の順に選択します。
6. パイプラインには、コンプライアンスの理由でクレジットカード情報をサニタイズする Lambda 関数が設定されています。

   保存済みの JSON ペイロードにクレジットカード情報が含まれていないことを確認するために、ファイルを解凍します。

ステップ 3: イベントの検索および分析パイプラインの実行を検証するには

1. OpenSearch サービスコンソールにサインインします。
2. ナビゲーションパネルの [My domains] で、`serverl-analyt` というプレフィックスが付いたドメインを選択します。

3. パイプラインには、数値一致条件を設定する Amazon SNS サブスクリプションフィルターポリシーが設定されています。

   金額が 100 USD を超える注文を参照するためにイベントのインデックスが作成されていることを確認するには、[serverl-analyt-***abcdefgh1ijk***] ページで、[インデックス]、[checkout_events] の順に選択します。

ステップ 4: イベントの再生パイプラインの実行を検証するには

1. Amazon SQS コンソールにサインインします。
2. キューのリストで、`serverlessrepo-fork-example` を検索して `ReplayQueue` を選択します。
3. [メッセージの送信と受信] を選択します。
4. fork-example-ecommerce「-***my-app*** でのメッセージの送受信」で、次の操作を行います。ReplayP -ReplayQueue-***123ABCD4E5F6*** ダイアログボックスで、メッセージにポーリングを選択します。
5. イベントがキューに入っていることを確認するには、キューに表示されているメッセージの横にある [詳細] を選択します。

ステップ 4: 復旧のために問題をシミュレートしてイベントを再生するには

ステップ 1: シミュレートした問題を有効にして別の API リクエストを送信するには

1. AWS Lambda コンソール にサインインします。
2. ナビゲーションパネルで [関数] を選択します。
3. `serverlessrepo-fork-example` を検索して `CheckoutFunction` を選択します。
4. fork-example-ecommerce-***my-app*** -CheckoutFunction-***ABCDEF*** ... ページの環境変数セクションで、BUG_ENABLED 変数を true に設定し、保存を選択します。
5. 次の JSON を `test_event_2.json` という名前のファイルにコピーします。

```
{
    "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. API エンドポイントに HTTPS リクエストを送信するには、curl コマンドを実行してサンプルイベントペイロードを入力として渡します。次に例を示します。

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

API は、次の空のレスポンスを返し、実行が成功したことを示します。

```
{ }
```

ステップ 2: シミュレートしたデータの破損を検証するには

1. Amazon DynamoDB コンソールにサインインします。
2. ナビゲーションパネルで、[テーブル] を選択します。
3. `serverlessrepo-fork-example` を検索して `CheckoutTable` を選択します。
4. テーブルの詳細ページで [項目] を選択し、作成済みの項目を選択します。

   保存済みの属性が表示されます。一部の属性は [CORRUPTED!] とマークされています。

ステップ 3: シミュレートした問題を無効にするには

1. AWS Lambda コンソール にサインインします。
2. ナビゲーションパネルで [関数] を選択します。
3. `serverlessrepo-fork-example` を検索して `CheckoutFunction` を選択します。
4. fork-example-ecommerce-***my-app*** -CheckoutFunction-***ABCDEF*** ... ページの「環境変数」セクションで、BUG_ENABLED 変数を false に設定し、「保存」を選択します。

ステップ 4: 再生を有効にして問題から復旧するには

1. AWS Lambda コンソールのナビゲーションパネルで、[関数] を選択します。
2. `serverlessrepo-fork-example` を検索して `ReplayFunction` を選択します。
3. [Designer] セクションを展開して [SQS] タイルを選択し、次に [SQS] セクションで [有効] を選択します。

   > **Note**
   >
   > Amazon SQS のイベントソーストリガーが有効になるまで約 1 分かかります。

4. [保存] を選択します。
5. 復旧された属性を表示するには、Amazon DynamoDB コンソールに戻ります。
6. 再生を無効にするには、AWS Lambda コンソールに戻り、Amazon SQS の `ReplayFunction` イベントソーストリガーを無効にします。

Amazon SNS トピックに AWS Event Fork Pipelines をサブスクライブする。

イベント駆動型アプリケーションの開発を迅速化するために、AWS Event Fork Pipelines により、イベント処理パイプラインを Amazon SNS トピックへサブスクライブできます。AWSEvent Fork Pipelines は、AWS サーバーレスアプリケーションモデル (AWS SAM) に基づいた、オープンソースのネストされたアプリケーションのスイートであり、AWS Event Fork Pipelines スイート ([Show apps that create custom IAM roles or resource policies] (カスタム IAM ロールまたはリソースポリシーを作成するアプリケーションの表示) を選択) から AWS アカウントへ直接デプロイできます。詳細については、「AWS Event Fork Pipelines の仕組み」を参照してください。

このセクションでは、AWS Management Console を使用してパイプラインをデプロイし、Amazon SNS トピックに AWS Event Fork Pipelines をサブスクライブする方法を説明します。開始する前に、Amazon SNS トピックを作成します。

パイプラインを構成するリソースを削除するには、AWS Lambdaコンソールの のアプリケーションページでパイプラインを見つけ、SAM テンプレートセクション を展開し、CloudFormationスタック を選択し、その他のアクション 、スタック の削除 を選択します。

トピック

- イベントのストレージおよびバックアップパイプラインをデプロイしてサブスクライブするには
- イベントの検索および分析パイプラインをデプロイしてサブスクライブするには
- イベントの再生パイプラインをデプロイしてサブスクライブするには

イベントのストレージおよびバックアップパイプラインをデプロイしてサブスクライブするには

イベントのアーカイブと分析のために、Amazon SNS は Amazon Data Firehose とのネイティブ統合の使用を推奨するようになりました。Firehose 配信ストリームを SNS トピックにサブスクライブできます。これにより、Amazon Simple Storage Service (Amazon S3) バケット、Amazon Redshift テーブル、Amazon OpenSearch Service (OpenSearch サービス) などのアーカイブエンドポイントと分析エンドポイントに通知を送信できます。Firehose 配信ストリームで Amazon SNS を使用することは、 AWS Lambda関数を使用する必要のない、フルマネージド型のコードレスソリューションです。詳細については、「Firehose 配信ストリームへのファンアウト」を参照してください。

このページでは、イベントのストレージおよびバックアップパイプラインをデプロイして Amazon SNS トピックにサブスクライブする方法を説明します。このプロセスでは、パイプラインに関連付けられた AWS SAM テンプレートを自動的に AWS CloudFormation スタックに変換し、このスタックを AWS アカウント 内にデプロイします。また、このプロセスでは、イベントのストレージおよびバックアップパイプラインを構成する、以下のようなリソースのセットを作成して設定します。

- Amazon SQS キュー
- Lambda 関数
- Firehose 配信ストリーム
- Amazon S3 バックアップバケット

S3 バケットを送信先としてストリームを設定する方法の詳細については、「Amazon Data Firehose API リファレンスS3DestinationConfiguration」の「」を参照してください。

イベントの変換、およびイベントバッファリング、イベント圧縮、イベント暗号化の設定の詳細については、「Amazon Data Firehose デベロッパーガイド」の「Amazon Data Firehose 配信ストリームの作成」を参照してください。

イベントのフィルター処理の詳細については、このガイドの「Amazon SNS サブスクリプションフィルターポリシー」を参照してください。

1. AWS Lambda コンソール にサインインします。
2. ナビゲーションパネルで [関数] を選択し、[関数の作成] を選択します。
3. [関数の作成] ページで、次の操作を実行します。
   a. [サーバーレスアプリケーションリポジトリの参照]、[パブリックアプリケーション]、[カスタム IAM ロールまたはリソースポリシーを作成するアプリケーションの表示] を選択します。
   b. `fork-event-storage-backup-pipeline` を検索し、このアプリケーションを選択します。
4. fork-event-storage-backupパイプラインページで、次の操作を行います。
   a. [アプリケーション設定] セクションで、[アプリケーション名] に名前 (`my-app-backup` など) を入力します。

> **Note**
>
> - 名前は、デプロイごとに一意にする必要があります。アプリケーション名を再利用すると、デプロイでは、スタックを新規作成せずに、以前にデプロイした AWS CloudFormation スタックを単に更新します。

b. （オプション）にはBucketArn、受信イベントがロードされる S3 バケットの ARN を入力します。値を入力しないと、新しい S3 バケットが AWS アカウントに作成されます。

c. （オプション）にはDataTransformationFunctionArn、受信イベントが変換される Lambda 関数の ARN を入力します。値を入力しないと、データ変換は無効になります。

d. （オプション）アプリケーションの Lambda 関数を実行するには、次のいずれかLogLevelの設定を入力します。

- `DEBUG`
- `ERROR`
- `INFO` (デフォルト)
- `WARNING`

e. にはTopicArn、フォークパイプラインのこのインスタンスをサブスクライブする Amazon SNS トピックの ARN を入力します。

f. （オプション）StreamBufferingIntervalInSecondsおよび StreamBufferingSizeInMBs、受信イベントのバッファリングを設定するための値を入力します。値を入力しないと、300 秒と 5 MB が使用されます。

g. （オプション）受信イベントを圧縮するには、次のいずれかStreamCompressionFormatの設定を入力します。

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (デフォルト)
- `ZIP`

h. （オプション）にStreamPrefix、S3 バックアップバケットに保存されているファイルの名前を付ける文字列プレフィックスを入力します。値を入力しないと、プレフィックスは使用されません。

i. （オプション) にはSubscriptionFilterPolicy、受信イベントのフィルタリングに使用する Amazon SNS サブスクリプションフィルターポリシーを JSON 形式で入力します。フィルターポリシーは、 OpenSearch サービスインデックスでインデックスを作成するイベントを決定します。値を入力しないと、フィルター処理は使用されません (すべてのイベントにインデックスが作成されます)。

j. （オプション) にはSubscriptionFilterPolicyScope、文字列 `MessageBody`または `MessageAttributes` を入力してペイロードベースまたは属性ベースのメッセージフィルタリングを有効にします。

k. [I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.] を選択して、[デプロイ] を選択します。

[***my-app*** のデプロイステータス] ページに、Lambda によって [Your application is being deployed] ステータスが表示されます。

[リソース] セクションで、AWS CloudFormation によってスタックの作成が開始され、各リソースの [CREATE_IN_PROGRESS] ステータスが表示されます。プロセスが完了すると、AWS CloudFormationに [CREATE_COMPLETE]ステータスが表示されます。

デプロイが完了すると、Lambda によって [アプリケーションはデプロイ済みです] ステータスが表示されます。

Amazon SNS トピックに発行されたメッセージは、イベントのストレージおよびバックアップパイプラインによってプロビジョニングされた S3 バックアップバケットに自動的に保存されます。

イベントの検索および分析パイプラインをデプロイしてサブスクライブするには

イベントのアーカイブと分析のために、Amazon SNS は Amazon Data Firehose とのネイティブ統合の使用を推奨するようになりました。Firehose 配信ストリームを SNS トピックにサブスクライブできます。これにより、Amazon Simple Storage Service (Amazon S3) バケット、Amazon Redshift テーブル、Amazon OpenSearch Service (OpenSearch サービス) などのアーカイブエンドポイントと分析エンドポイントに通知を送信できます。Firehose 配信ストリームで Amazon SNS を使用することは、 AWS Lambda関数を使用する必要のない、フルマネージド型のコードレスソリューションです。詳細については、「Firehose 配信ストリームへのファンアウト」を参照してください。

このページでは、イベントの検索および分析パイプラインをデプロイして Amazon SNS トピックにサブスクライブする方法を示します。このプロセスでは、パイプラインに関連付けられた AWS SAM テンプレートを自動的に AWS CloudFormation スタックに変換し、このスタックを AWS アカウント 内にデプロイします。また、このプロセスでは、イベントの検索および分析パイプラインを構成する、以下のようなリソースのセットを作成して設定します。

- Amazon SQS キュー
- Lambda 関数
- Firehose 配信ストリーム
- Amazon OpenSearch Service ドメイン
- Amazon S3 配信不能バケット

インデックスを送信先としてストリームを設定する方法の詳細については、Amazon Data Firehose API リファレンスElasticsearchDestinationConfigurationの「」を参照してください。

イベントの変換、およびイベントバッファリング、イベント圧縮、イベント暗号化の設定の詳細については、「Amazon Data Firehose デベロッパーガイド」の「Amazon Data Firehose 配信ストリームの作成」を参照してください。

イベントのフィルター処理の詳細については、このガイドの「Amazon SNS サブスクリプションフィルターポリシー」を参照してください。

1. AWS Lambda コンソール にサインインします。
2. ナビゲーションパネルで [関数] を選択し、[関数の作成] を選択します。
3. [関数の作成] ページで、次の操作を実行します。
   a. [サーバーレスアプリケーションリポジトリの参照]、[パブリックアプリケーション]、[カスタム IAM ロールまたはリソースポリシーを作成するアプリケーションの表示] を選択します。
   b. `fork-event-search-analytics-pipeline` を検索し、このアプリケーションを選択します。
4. fork-event-search-analyticsパイプラインページで、次の操作を行います。
   a. [アプリケーション設定] セクションで、[アプリケーション名] に名前 (`my-app-search` など) を入力します。

> **Note**
>
> 名前は、デプロイごとに一意にする必要があります。アプリケーション名を再利用すると、デプロイでは、スタックを新規作成せずに、以前にデプロイした AWS CloudFormation スタックを単に更新します。

b. （オプション）にはDataTransformationFunctionArn、受信イベントの変換に使用される Lambda 関数の ARN を入力します。値を入力しないと、データ変換は無効になります。

c. （オプション）アプリケーションの Lambda 関数を実行するには、次のいずれかLogLevelの設定を入力します。

- `DEBUG`
- `ERROR`
- `INFO` (デフォルト)
- `WARNING`

d. （オプション）にはSearchDomainArn、必要なコンピューティング機能とストレージ機能を設定するクラスターである OpenSearch サービスドメインの ARN を入力します。値を入力しないと、新しいドメインがデフォルト設定で作成されます。

e. にはTopicArn、フォークパイプラインのこのインスタンスをサブスクライブする Amazon SNS トピックの ARN を入力します。

f. にはSearchIndexName、イベントの検索と分析のサービス OpenSearch インデックスの名前を入力します。

> **Note**
>
> インデックス名には、次の制限が適用されます。
>
> - 大文字を含めることはできません
> - 次の文字を含めることはできません: `\ / * ? " < > | ` , #`
> - 次の文字で始めることはできません: `- + _`
> - 次の形式にすることはできません: `. ..`
> - 80 文字より長くすることはできません
> - 255 バイトより長くすることはできません
> - コロンを含めることはできません（OpenSearch サービス 7.0 から）

g. （オプション) OpenSearch サービスインデックスのローテーション期間中は、次のいずれかSearchIndexRotationPeriodの設定を入力します。

- `NoRotation` (デフォルト)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

インデックスのローテーションでは、インデックス名にタイムスタンプを付加し、古いデータの有効期限切れをわかりやすくします。

h. にはSearchTypeName、インデックス内のイベントを整理するための OpenSearch サービスタイプの名前を入力します。

> **Note**
>
> - OpenSearch サービスタイプ名には任意の文字 (null バイトを除く) を含めることができますが、 で始めることはできません_。
> - OpenSearch サービス 6.x の場合、インデックスごとに 1 つのタイプしか存在できません。既に別のタイプを持つ既存のインデックスに新しいタイプを指定すると、Firehose はランタイムエラーを返します。

i. （オプション) StreamBufferingIntervalInSecondsおよび StreamBufferingSizeInMBs、受信イベントのバッファリングを設定するための値を入力します。値を入力しないと、300 秒と 5 MB が使用されます。

j. （オプション) 受信イベントを圧縮するには、次のいずれかStreamCompressionFormatの設定を入力します。

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (デフォルト)
- `ZIP`

k. （オプション) にはStreamPrefix、S3 デッドレターバケットに保存されているファイルの名前を付ける文字列プレフィックスを入力します。値を入力しないと、プレフィックスは使用されません。

l. （オプション) にはStreamRetryDurationInSecons、Firehose が OpenSearch サービスインデックスにイベントのインデックスを作成できない場合の再試行期間を入力します。値を入力しないと、300 秒が使用されます。

m. （オプション) にはSubscriptionFilterPolicy、受信イベントのフィルタリングに使用する Amazon SNS サブスクリプションフィルターポリシーを JSON 形式で入力します。フィルターポリシーは、 OpenSearch サービスインデックスでインデックスを作成するイベントを決定します。値を入力しないと、フィルター処理は使用されません (すべてのイベントにインデックスが作成されます)。

n. [I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.] を選択して、[デプロイ] を選択します。

のデプロイステータス ***my-app-search*** ページに、Lambda はアプリケーションをデプロイ中のステータスを表示します。

[リソース] セクションで、AWS CloudFormation によってスタックの作成が開始され、各リソースの [CREATE_IN_PROGRESS] ステータスが表示されます。プロセスが完了すると、AWS CloudFormationに [CREATE_COMPLETE]ステータスが表示されます。

デプロイが完了すると、Lambda によって [アプリケーションはデプロイ済みです] ステータスが表示されます。

Amazon SNS トピックに発行されたメッセージは、イベント検索および分析パイプラインによってプロビジョニングされた OpenSearch サービスインデックスに自動的にインデックス作成されます。パイプラインでイベントのインデックスを作成できない場合、イベントは S3 配信不能バケットに保存されます。

イベントの再生パイプラインをデプロイしてサブスクライブするには

このページでは、[イベントの再生パイプライン] をデプロイして Amazon SNS トピックにサブスクライブする方法を示します。このプロセスでは、パイプラインに関連付けられた AWS SAM テンプレートを自動的に AWS CloudFormation スタックに変換し、このスタックを AWS アカウント 内にデプロイします。また、このプロセスでは、イベントの再生パイプラインを構成する Amazon SQS キューや Lambda 関数などのリソースのセットを作成して設定します。

イベントのフィルター処理の詳細については、このガイドの「Amazon SNS サブスクリプションフィルターポリシー」を参照してください。

1. AWS Lambda コンソール にサインインします。
2. ナビゲーションパネルで [関数] を選択し、[関数の作成] を選択します。
3. [関数の作成] ページで、次の操作を実行します。

   a. [サーバーレスアプリケーションリポジトリの参照]、[パブリックアプリケーション]、[カスタム IAM ロールまたはリソースポリシーを作成するアプリケーションの表示] を選択します。

   b. `fork-event-replay-pipeline` を検索し、このアプリケーションを選択します。
4. [fork-event-replay-pipeline] ページで以下の操作を実行します。

   a. [アプリケーション設定] セクションで、[アプリケーション名] に名前 (`my-app-replay` など) を入力します。

   > **Note**
   >
   > 名前は、デプロイごとに一意にする必要があります。アプリケーション名を再利用すると、デプロイでは、スタックを新規作成せずに、以前にデプロイした AWS CloudFormation スタックを単に更新します。

   b. (オプション) アプリケーションの Lambda 関数を実行するには、次のいずれかLogLevelの設定を入力します。

   - `DEBUG`
   - `ERROR`
   - `INFO` (デフォルト)
   - `WARNING`

   c. (オプション) にはReplayQueueRetentionPeriodInSeconds、Amazon SQS 再生キューがメッセージを保持する時間を秒単位で入力します。値を入力しないと、1,209,600 秒 (14 日間) が使用されます。

   d. にはTopicArn、フォークパイプラインのこのインスタンスをサブスクライブする Amazon SNS トピックの ARN を入力します。

   e. にはDestinationQueueName、Lambda 再生関数がメッセージを転送している Amazon SQS キューの名前を入力します。

f. （オプション) にはSubscriptionFilterPolicy、受信イベントのフィルタリングに使用する Amazon SNS サブスクリプションフィルターポリシーを JSON 形式で入力します。フィルターポリシーは、再生用にバッファ処理するイベントを決定します。値を入力しないと、フィルター処理は使用されません (すべてのイベントが再生用にバッファ処理されます)。

g. [I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.] を選択して、[デプロイ] を選択します。

のデプロイステータス ***my-app-replay*** ページに、Lambda はアプリケーションをデプロイ中のステータスを表示します。

[リソース] セクションで、AWS CloudFormation によってスタックの作成が開始され、各リソースの [CREATE_IN_PROGRESS] ステータスが表示されます。プロセスが完了すると、AWS CloudFormationに [CREATE_COMPLETE]ステータスが表示されます。

デプロイが完了すると、Lambda によって [アプリケーションはデプロイ済みです] ステータスが表示されます。

Amazon SNS トピックに発行されたメッセージは、イベントの再生パイプラインによってプロビジョニングされた Amazon SQS キューで自動的に再生用にバッファ処理されます。

> **Note**
>
> 再生は、デフォルトでは無効になります。再生を有効にするには、Lambda コンソールで関数のページに移動し、[Designer] セクションを展開して [SQS] タイルを選択します。次に [SQS] セクションで [有効] を選択します。

Amazon SNS での Amazon EventBridge スケジューラの使用

Amazon EventBridge スケジューラはサーバーレススケジューラで、一元化されたマネージドサービスからタスクを作成、実行、管理できます。EventBridge スケジューラでは、繰り返しのパターンに Cron やレート式を使ってスケジュールを作成したり、1回限りの呼び出しを設定したりできます。配信時間枠の柔軟な設定、再試行制限の定義、失敗した API 呼び出しの最大保持時間の設定を行うことができます。

このページでは、EventBridge スケジューラを使用してスケジュールに基づき Amazon SNS トピックからメッセージを発行する方法について説明します。

トピック

実行ロールを設定する

新しいスケジュールを作成する場合、EventBridge スケジューラにはユーザーに代わってターゲット API オペレーションを呼び出すアクセス許可が必要です。実行ロールを使用して、これらのアクセス許可を EventBridge スケジューラに付与します。スケジュールの実行ロールにアタッチするアクセス許可ポリシーによって、必要なアクセス許可が定義されます。これらのアクセス許可は、EventBridge スケジューラが呼び出すターゲット API によって異なります。

次の手順のように EventBridge スケジューラコンソールを使用してスケジュールを作成すると、EventBridge スケジューラは選択したターゲットに基づき実行ロールを自動的に設定します。EventBridge スケジューラ SDK、AWS CLI、または AWS CloudFormation のいずれかを使用してスケジュールを作成する場合、EventBridge スケジューラがターゲットを呼び出すために必要なアクセス許可を付与する既存の実行ロールが必要です。スケジュールに合わせて実行ロールを手動で設定する方法についての詳細は、「EventBridge スケジューラユーザーガイド」の「実行ロールを設定する」を参照してください。

新しいスケジュールを作成する

コンソールを使用してスケジュールを作成するには

1. Amazon EventBridge スケジューラコンソール (https://console.aws.amazon.com/scheduler/home) を開きます。
2. [スケジュール] ページで、[スケジュールを作成] を選択します。
3. [スケジュールの詳細を指定] ページの [スケジュールの名前と説明] セクションで、次を実行します。

    a. [スケジュール名] で、スケジュールの名前を入力します。例えば、**`MyTestSchedule`** です。

    b. (オプション) [説明] で、スケジュールの説明を入力します。例えば、**`My first schedule`** です。

c. [スケジュールグループ] で、ドロップダウンリストからスケジュールグループを選択します。グループがない場合は、[デフォルト] を選択します。スケジュールグループを作成するには、[独自のスケジュールを作成] を選択します。

スケジュールグループを使用して、スケジュールのグループにタグを追加します。

4. • スケジュールオプションを選択します。

頻度	手順	
[1 回限りのスケジュール] 1 回限りのスケジュールは、指定した日時に 1 回だけターゲットを呼び出します。	[日付と時刻] で、次を実行します。 • 有効な日付を YYYY/MM/DD 形式で入力します。 • タイムスタンプを 24 時間 (hh:mm) 形式で入力します。 • [タイムゾーン] で、タイムゾーンを選択します。	
[繰り返しのスケジュール] 繰り返しのスケジュールは、cron 式またはレート式を使用して指定したレートでターゲットを呼び出します。	a. [スケジュールの種類] では、次のいずれかを実行します。 • Cron 式を使用してスケジュールを定義するには、[cron ベースのスケジュール] を選択して Cron 式を入力します。 • Rate 式を使用してスケジュールを定義するには、[rate ベースのスケジュール] を選択して Rate 式を入力します。	

頻度	手順	
	cron およびレート式の詳細については、「Amazon EventBridge スケジューラユーザーガイド」の「EventBridge スケジューラのスケジュールタイプ」を参照してください。 b. [フレックスタイムウィンドウ] で、[オフ] を選択してオプションをオフにするか、事前定義された時間枠のいずれかを選択します。例えば、[15 分] を選択し、1 時間に 1 回ターゲットを呼び出す繰り返しのスケジュールを設定した場合、スケジュールは毎時の開始後 15 分以内に実行されます。	

5. (オプション) 前のステップで [定期的なスケジュール] を選択した場合は、[時間枠] セクションで次を実行します。

   a. [タイムゾーン] で、タイムゾーンを選択します。

   b. [開始日時] で、有効な日付を YYYY/MM/DD 形式で入力してから、タイムスタンプを 24 時間 (hh:mm) 形式で指定します。

   c. [終了日時] で、有効な日付を YYYY/MM/DD 形式で入力してから、タイムスタンプを 24 時間 (hh:mm) 形式で指定します。

6. [Next] (次へ) をクリックします。

7. [ターゲットを選択] ページで、EventBridge スケジューラが呼び出す AWS API オペレーションを選択します。

a. [Amazon SNS 発行] を選択します。

b. [発行] セクションで、SNS トピックを選択するか、[新しい SNS トピックを作成] を選択します。

c. (オプション) JSON ペイロードを入力します。ペイロードを入力しない場合、EventBridge スケジューラは空のイベントを使用して関数を呼び出します。

8. [Next] (次へ) をクリックします。

9. [Settings] (設定) ページで、以下の操作を行います。

a. スケジュールをオンにするには、[スケジュールの状態] で [スケジュールを有効にする] をオンに切り替えます。

b. スケジュールの再試行ポリシーを設定するには、[再試行ポリシーとデッドレターキュー (DLQ)] で次を実行します。

- [再試行] を切り替えてオンにします。
- [イベントの最大有効期間] で、EventBridge スケジューラが未処理のイベントを保持しなければならない最大の [時間] と [分] を入力します。
- 最大 24 時間です。
- [最大再試行回数] で、ターゲットがエラーを返した場合に EventBridge スケジューラがスケジュールを再試行する最大回数を入力します。

  再試行の最大値は 185 です。

再試行ポリシーを使用すると、スケジュールがそのターゲットの呼び出しに失敗した場合、EventBridge スケジューラはスケジュールを再実行します。設定されている場合は、スケジュールの最大保持時間と再試行を設定する必要があります。

c. EventBridge スケジューラが未配信のイベントを保存する場所を選択します。

[デッドレターキュー (DLQ)] オプション	手順	
保存しない	[None] を選択します。	
スケジュールを作成しようとしている同じ AWS アカ	a. [自分の AWS アカウント の Amazon SQS	

[デッドレターキュー (DLQ)] オプション	手順
ウント にイベントを保存する	キューを DLQ として選択] を選択します。 b. Amazon SQS キューの Amazon リソースネーム (ARN) を選択します。
スケジュールを作成しようとしているのとは別の AWS アカウント にイベントを保存する	a. [他の AWS アカウント の Amazon SQS キューを DLQ として指定] を選択します。 b. Amazon SQS キューの Amazon リソースネーム (ARN) を入力します。

d. カスタマーマネージドキーを使用してターゲットの入力を暗号化するには、[暗号化] で [暗号化設定をカスタマイズする (高度)] を選択します。

このオプションを選択した場合は、既存の KMS キー ARN を入力するか、[AWS KMS key を作成] を選択して AWS KMS コンソールに移動します。EventBridge スケジューラが保管中のデータを暗号化する方法の詳細については、「Amazon EventBridge スケジューラユーザーガイド」の「保管中の暗号化」を参照してください。

e. EventBridge スケジューラに新しい実行ロールを作成させるには、[このスケジュールの新しいロールを作成] を選択します。その後、[ロール名] で名前を入力します。このオプションを選択すると、EventBridge スケジューラは、テンプレート化されたターゲットに必要な許可をロールにアタッチします。

10. [Next] (次へ) をクリックします。
11. [スケジュールの確認と作成] ページで、スケジュールの詳細を確認します。各セクションで、そのステップに戻って詳細を編集するには、[編集] を選択します。
12. [スケジュールを作成] を選択します。

[スケジュール] ページで、新規および既存のスケジュールのリストを表示できます。[ステータス] 列で、新しいスケジュールが [有効] になっていることを確認します。

関連リソース

EventBridge スケジューラに関する詳細については、次を参照してください。

- EventBridge スケジューラユーザーガイド
- EventBridge スキーマ API リファレンス
- EventBridge Scheduler Pricing

Amazon SNS を使用した Application-to-Person (A2P) メッセージング

このセクションでは、モバイルアプリケーション、携帯電話番号、E メールアドレスなどの受信者へのユーザー通知に Amazon SNS を使用する方法について説明します。

トピック

モバイルテキストメッセージング (SMS)

Amazon SNS を使用して、SMS 対応デバイスにテキストメッセージ (SMS メッセージ) を送信できます。電話番号をトピックにサブスクライブし、トピックへメッセージを送信することにより、電話番号へメッセージを直接送信または、一度に複数の電話番号にメッセージを送信できます。

AWS アカウントに対する SMS プリファレンスを設定して、目的のユースケースと予算に対して SMS 配信をカスタマイズできます。例えば、ユーザーは、メッセージがコストに対して、または確実な配信に対して最適化されるかどうかを選択できます。また、個別のメッセージ配信の使用量の限度、および AWS アカウント の毎月の使用料金の限度を指定できます。

現地の法律および規制により義務付けられている場合 (米国およびカナダなど)、SMS の受信者はオプトアウトができます。これは、ユーザーの AWS アカウント からの SMS メッセージの受信の停止を選択することを意味します。受信者がオプトアウトした後、ユーザーは、制限付きで、再度電話番号をオプトインし、メッセージの送信を再開できます。

Amazon SNS は、複数のリージョンで SMS メッセージングをサポートしているので、ユーザーは、200 以上の国とリージョンにメッセージを送信できます。詳細については、「サポートされている国と地域」を参照してください。

トピック

- SMS メッセージプリファレンスを設定する
- SMS メッセージの送信
- SMS のアクティビティをモニタリングする
- 電話番号および SMS サブスクリプションを管理する
- サポートされている国と地域
- SMS ベストプラクティス

SMS サンドボックス

Amazon SNS を使用して SMS メッセージを送信し始めると、AWS アカウントの所在地は SMS サンドボックス内になります。SMS サンドボックスは、SMS 送信者としての評判を損なうことなく、Amazon SNS 機能を試すための安全な環境を提供します。アカウントは SMS サンドボックスにありますが、Amazon SNS のすべての機能を使用することができます。ただし、以下の制限があります。

- SMS メッセージは、認証済みの送信先電話番号にのみ送信できます。
- 最大 10 個の認証済みの送信先電話番号を持つことができます。
- 送信先の電話番号を削除できるのは、認証または最後の認証の試行から 24 時間以上経過後に限られます。

アカウントがサンドボックスから移動されると、これらの制限は解除され、SMS メッセージを任意の受信者に送信できます。

トピック

- SMS サンドボックスでの電話番号の追加と確認
- SMS サンドボックスで電話番号を削除する
- SMS サンドボックス外への移動

SMS サンドボックスでの電話番号の追加と確認

AWS アカウントが SMS サンドボックス にある間に SMS メッセージの送信を開始するには、発信元 ID を作成し、送信先の電話番号を追加して、検証します。

> **Note**
>
> SMS サンドボックス内にないアカウントと同様に、一部の国やリージョンの受信者に SMS メッセージを送信するには、事前に送信元アイデンティティが必要になります。詳細については、「サポートされている国と地域」を参照してください。
> 送信元 ID には送信者 ID と、さまざまなタイプの送信元番号が含まれます。既存の送信元番号を表示するには、[Amazon SNS コンソール] で、[送信元番号] を選択します。現在、送信者 ID はこのリストに表示されていません。

送信先電話番号を追加および確認するには

1. Amazon SNS コンソールにサインインします。
2. 電話番号の送信元アイデンティティを作成します。
3. コンソールメニューで、SMS メッセージングをサポートしている AWS リージョンを選択します。
4. ナビゲーションペインで、[テキストメッセージング (SMS)] を選択します。
5. リポジトリの「モバイルテキストメッセージング (SMS)」ページで、[サンドボックスの送信先電話番号] の [電話番号を追加] を選択します。
6. [送信先の詳細] で、国番号と電話番号を入力し、認証メッセージに使用する言語を指定し、[電話番号を追加] を選択します。

   Amazon SNS は送信先電話番号に、ワンタイムパスワード (OTP) を送信します。送信先電話番号が 15 分以内に OTP を受信しない場合は、[認証コードを再送信する] を選択します。OTP は、24 時間ごとに最大 5 回まで同じ送信先電話番号に送信できます。
7. [確認コード] のボックスに、送信先電話番号に送信される OTP を入力し、[電話番号を認証する] を選択します。

   送信先電話番号とその認証ステータスが、サンドボックスの送信先電話番号セクションに表示されます。認証ステータスが保留の場合、認証は失敗しています。これは、例えば、入力した電話番号に国コードが含まれていなかった場合などに発生します。保留中または認証済みの送信先電話番号を削除できるのは、認証または最後の認証の試行から 24 時間以上経過後に限られます。
8. この送信先電話番号を使用するリージョンごとに、上記のステップを繰り返します。

OTP テキストの受信なしのトラブルシューティング

電話番号が OTP テキストを受信できない可能性がある一般的な問題をトラブルシューティングします。

- Amazon SNS SMS の使用制限： AWS アカウント が SMS メッセージを送信するための使用制限を超えた場合、OTP テキストを含むそれ以上のメッセージは、制限が引き上げられるか、請求の問題が解決されるまで配信されない場合があります。
- SMS 通知をオプトインしていない電話番号： 一部の国または地域では、受信者はショートコードから SMS メッセージを受信するようにオプトインする必要があります。ショートコードは OTP テキストによく使用されます。受信者の電話番号がオプトインされていない場合、OTP テキストは受信されません。
- 通信事業者の制限またはフィルタリング： 一部の携帯電話通信事業者では、OTP テキストを含む特定のタイプの SMS メッセージの配信を妨げる制限またはフィルタリングメカニズムがある場合があります。これは、セキュリティポリシーまたは通信事業者が実装したスパム対策が原因である可能性があります。
- 無効または誤った電話番号： 受信者から提供された電話番号が正しくないか無効である場合、OTP テキストは配信されません。
- ネットワークの問題： 一時的なネットワークの問題または停止により、OTP テキストを含む SMS メッセージが受信者の電話に配信されない場合があります。
- 配信の遅延： ネットワーク輻輳やその他の要因により、配信が遅延することがあります。OTP テキストは最終的には配信される可能性がありますが、予想される期間を超えて遅れる可能性があります。
- アカウントの停止または終了： サービス AWS 条件の非侵害や違反 AWS アカウントなど、 に問題がある場合、OTP テキストを含む Amazon SNS メッセージング機能が停止または終了されることがあります。

SMS サンドボックスで電話番号を削除する

SMS サンドボックスで、保留中または認証済みの送信先電話番号を削除できます。

SMS サンドボックスから送信先の電話番号を削除するには

1. 電話番号を認証を行ってから、または最後の認証の試行から 24 時間待ってください。
2. Amazon SNS コンソールにサインインします。

3. コンソールメニューで、送信先電話番号を追加した、AWS SMS メッセージングをサポートしているリージョン を選択します。
4. ナビゲーションペインで、[テキストメッセージング (SMS)]を選択します。
5. リポジトリの「モバイルテキストメッセージング (SMS)」ページで、[サンドボックスの送信先電話番号] を選択し、次に [電話番号を追加] を選択します。
6. 送信先電話番号の削除を確定するには、「**`delete me`**」と入力し、[削除] を選択します。

   送信先電話番号を認証してから、または認証を試みてから 24 時間以上経過していれば、送信先電話番号は削除され、Amazon SNS によって送信先電話番号のリストが更新されます。
7. 追加したけれどもう使わない送信先電話番号を追加したリージョンごとに、上記のステップを繰り返します。

SMS サンドボックス外への移動

SMS サンドボックス AWS アカウント から を移動するには、まず送信先の電話番号を追加、検証、テストする必要があります。次に、 を使用してケースを作成する必要があります AWS Support。

AWS アカウントが SMS サンドボックスの外に移動されるようにリクエストするには

1. 電話番号を確認する

   a. AWS アカウント が SMS サンドボックスにある間に、Amazon SNS コンソール を開きます。
   b. ナビゲーションペインのモバイルで、テキストメッセージ (SMS) を選択します。
   c. サンドボックスの送信先電話番号セクションで、1 つ以上の送信先電話番号を追加して確認します。この検証により、メッセージを正常に送受信できるようになります。
2. SMS 発行のテスト

   - 少なくとも 1 つの検証済み電話番号に対してメッセージを送受信できることを確認します。SMS メッセージの発行方法の詳細については、「」を参照してください携帯電話に発行する。
3. サンドボックスの編集を開始する

   - 「Amazon SNS コンソールのモバイルテキストメッセージング (SMS)」のページの [アカウント情報] で、[SMS サンドボックスを終了する] を選択します。。このアクションにより、Amazon サポートセンターにリダイレクトされ、Service Quotas increase オプションを選択してサポートケースが自動的に作成されます。

4. フォームへの入力

   - Service Quotas increase のサポートフォームで、次の操作を行います。

     i. サービスとして SNS テキストメッセージングを選択します。

     ii. SMS メッセージの送信元となるウェブサイト URL またはアプリ名を指定します。

     iii. 送信するメッセージのタイプを指定します: ワンタイムパスワード、プロモーション用、またはトランザクション用。

     iv. SMS メッセージを送信AWS リージョンする を選択します。

     v. SMS メッセージを送信する予定の国または地域を一覧表示します。

     vi. お客様がメッセージの受信をオプトインする方法を説明します。

     vii. 使用する予定のメッセージテンプレートをすべて含めます。

5. クォータとリージョンを指定する

   - [リクエスト]で、以下の操作を行います。

     i. を移動AWS リージョンする を選択します AWS アカウント。

     ii. リソースタイプ の一般的な制限を選択します。

     iii. クォータ で SMS サンドボックスを終了する を選択します。

     iv. (オプション) 追加の引き上げやその他の調整をリクエストするには、別のリクエストを追加を選択し、必要な詳細を指定します。

     v. 新しいクォータ値 には、リクエストする上限を USD で入力します。

6. 追加の詳細

   a. ケースの説明で、リクエストに関連する追加の詳細を入力します。

   b. 「問い合わせオプション」で、希望する問い合わせ言語を選択します。

7. リクエストを送信する

   - 送信を選択して、 にリクエストを送信します AWS Support。

AWS Support チームは 24 時間以内にリクエストに最初のレスポンスを提供します。

迷惑なコンテンツや悪意のあるコンテンツを送信するためにシステムが悪用されないように、私たちは各リクエストを慎重に検討しています。可能であれば、24 時間以内にリクエストを承認します。ただし、追加情報が必要な場合は、お客様のリクエストの解決に時間がかかる場合があります。

お客様のユースケースが私たちのポリシーに一致しない場合は、リクエストを承認できない場合があります。

SMS メッセージの送信元アイデンティティ

Amazon SNS を使用して SMS メッセージを送信する場合、次のタイプの送信元アイデンティティを使用して受信者に対しお客様の身元証明ができます。

- 送信者 ID
- 送信元番号

> **Note**
>
> Amazon SNS SMS メッセージングは、Amazon Pinpoint が現在サポートされていないリージョンで使用できます。欧州 (ストックホルム)、中東 (バーレーン)、欧州 (パリ)、南米 (サンパウロ)、米国西部 (北カリフォルニア) で運用中の場合、米国東部 (バージニア北部) リージョンの Amazon Pinpoint コンソールを開き、10DLC の会社とキャンペーンを登録します。ただし、10DLC 番号の要求は行わないでください。代わりに、AWSService Quotas コンソールを使用して、そのリージョンの 10DLC 番号を要求しながら、サービス制限の増加ケースを作成します。送信元 ID をリクエストする方法の詳細については、「Amazon SNS で SMS メッセージングのサポートをリクエストする」を参照してください。

送信者 ID

送信者 ID は、SMS メッセージの送信者を識別する英字名です。送信者 ID を使用して SMS メッセージを送信し、送信者 ID 認証がサポートされているエリアに受信者がいる場合は、電話番号ではなく、送信者 ID が受信者のデバイスに表示されます。送信者 ID は、電話番号、ロングコード、ショートコードよりも、送信者に関する詳細な情報を SMS の受信者に示すことができます。

送信者 ID は、世界の一部の国や地域でサポートされています。一部の地域では、企業として個別のお客様に SMS メッセージを送信する場合は、事前に規制当局または業界団体に登録されている送信者 ID を使用する必要があります。送信者 ID をサポートしている、または必要な国や地域の詳細な一覧は、「サポートされている国と地域」を参照してください。

送信者 ID の使用には追加料金はかかりません。ただし、送信者 ID 認証のサポートや要件は異なります。主要な市場 (例: カナダ、中国、米国) の中には、送信者 ID の使用がサポートされていないも

のがあります。一部の地域では、企業として個人のお客様に SMS メッセージを送信する場合は、規制当局または業界団体に事前登録されている送信者 ID を使用する必要があります。

> **Important**
>
> AWS は、送信者 ID を他者、他の会社、または他の製品を偽装するために使用する SMS スプーフィングを禁止します。所有しているブランドまたは商標を表す送信者 ID のみを使用してください。

利点

送信者 ID は、メッセージの送信者に関する詳細を受取人に示します。ショートコードやロングコードよりも、送信者 ID を使用した方が簡単にブランドのアイデンティティを確立することができます。送信者 ID の使用には追加料金はかかりません。

欠点

送信者 ID 認証のサポートや要件は、国やリージョンによって異なります。主要な市場 (例: カナダ、中国、米国) の中には、送信者 ID がサポートされていない場合があります。一部の工リアでは、送信者 ID を使用する前に、規制機関の事前承認を取得する必要があります。

国別の送信者 ID 登録

SMS メッセージングの送信者 ID を登録するには、AWS サポートでケースをオープンする必要があります。サポートケースを提起すると、AWS は、追加の必要なドキュメントを共有します。送信者 ID を登録する国に関して、以下の情報も指定する必要があります。

国名	メッセージの種類	形式の制限と要件	登録の要件
オーストラリア (AU)	トランザクションおよびプロモーション	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし • 送信者 ID は SMS を送信する会社の	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む)

国名	メッセージの種類	形式の制限と要件	登録の要件

		ブランド名である必要があります	• 会社の国
			• 会社の URL (アプリまたは会社のウェブサイトへのリンク)
			• 月間使用量の見積もり
			• ユースケースの説明、メッセージの目的
			• 明確でない場合は、会社名と送信者 ID の関係について説明してください
			• 会社の正式なビジネス/貿易ライセンス番号または VAT 番号
			• 送信する予定のメッセージテンプレート
			• 事業者登録ライセンス。次のような例があります (ただし、これらに限定されません)。
			  • Australian Business Number (ABN)
			  • Australian Company Number (ACN)

国名	メッセージの種類	形式の制限と要件	登録の要件
			• Australian Registered Body Number (ARBN) • Indigenous Corporation Number (ICN) • Letter of Authorization (LOA)

国名	メッセージの種類	形式の制限と要件	登録の要件
ベラルーシ (BY)	トランザクションメッセージのみ	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし • 送信者 ID は SMS を送信する会社のブランド名である必要があります	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください • 会社の正式なビジネス/貿易ライセンス番号または VAT 番号

国名	メッセージの種類	形式の制限と要件	登録の要件
			• 送信する予定のメッセージテンプレート

国名	メッセージの種類	形式の制限と要件	登録の要件
中国 (CN) 中国では送信者 ID の登録は必須ではありませんが、本文の先頭に署名を付けたコンテンツ/テンプレートの登録が必要です ([Amazon] など)。	次のキーワードは使用できません。 • Falung Gong • SB • Tiananmen Square 次のカテゴリからのメッセージ。 • クレジットカード • デジタル決済 (暗号通貨を含む) • 不正なトラフィック URL (フィッシングまたはスパム) • ギャンブル • 不適切なコンテンツ (成人向け、暴力、薬物、アルコール) • ローン • 整形手術 • 政治的 • 宗教的 • 株取引 • 仮想通貨 角括弧内の署名は、SMS メッセージ本文の先頭に付加する必要があります。中	• 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 会社名 • 会社の住所 • 都道府県 • 国 • 会社の電話番号 • 会社のウェブサイト • 月間使用量の見積もり • メッセージタイプ: プロモーション/トランザクション • ユースケースの説明 • 登録するテンプレート (送信する SMS は、コンプライアンスと配信の成功を保証するために提供されているテンプレートと異なるものであってはなりません)。例えば、[CompanyName] ワンタイムパスワードは {OTP} です。このコードは 10 分で期限切れになります。 • プロモーションコンテンツをトラ

国名	メッセージの種類	形式の制限と要件	登録の要件
	国への配信を成功させるには、メッセージコンテンツテンプレートにメッセージ署名を登録する必要があります。このメッセージ署名は、送信する各 SMS のメッセージ内容の先頭に付加する必要があります。SMS メッセージ本文の先頭に登録済みの署名を付加しないと、SMS がブロックまたはフィルタリングされる可能性があります。署名は SMS 本文の先頭に角括弧で囲んで付加する必要があります。		ンザクションメッセージタイプとして送信しないことの確認 • メッセージ署名。「署名形式の制限と要件」を参照してください。

国名	メッセージの種類	形式の制限と要件	登録の要件
エジプト (EG)	トランザクションメッセージのみ	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください • あなたの会社はエジプトにエンティティ/オフィスを持っていますか。それとも、これはエジプトに拠点を

国名	メッセージの種類	形式の制限と要件	登録の要件
			置かない会社が工ジプトに送っているのでしょうか。 • ユースケースがこの送信者 ID で送信されるすべてのメッセージを対象としていることを書面で確認する • 明確でない場合は、会社名と送信者 ID の関連情報を入力してください • 送信する予定のメッセージテンプレート
インド (IN)	トランザクションメッセージのみ	• 英数字 • 最大 6 文字 • スペースは使用できません • 特殊文字なし	「インドの送信者 ID 登録要件」を参照してください。

国名	メッセージの種類	形式の制限と要件	登録の要件
ヨルダン (JO)	トランザクションメッセージのみ	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください • ビジネス登録証明書 • 送信する予定のメッセージのタイプ (OTP、アラートなど)

国名	メッセージの種類	形式の制限と要件	登録の要件
			• この送信者 ID によって送信されるすべてのメッセージがユースケースに記述されていることの書面による確認 • 送信する予定のメッセージテンプレート

国名	メッセージの種類	形式の制限と要件	登録の要件
クウェート (KW)	トランザクションメッセージのみ	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください • 送信する予定のメッセージのタイプ (OTP、アラートなど) • この送信者 ID で送信されるすべての

国名	メッセージの種類	形式の制限と要件	登録の要件
			メッセージがユースケースに記述されていることの書面による確認 • 送信する予定のメッセージテンプレート

国名	メッセージの種類	形式の制限と要件	登録の要件
フィリピン (PH)	トランザクションメッセージのみ	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください • 送信する予定のメッセージのタイプ (OTP、アラートなど) • この送信者 ID によって送信される

国名	メッセージの種類	形式の制限と要件	登録の要件
			すべてのメッセージがユースケースに記述されていることの書面による確認 • 送信する予定のメッセージテンプレート

国名	メッセージの種類	形式の制限と要件	登録の要件
カタール (QA)	トランザクションメッセージのみ	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください • 送信される SMS のタイプ • (トランザクション/プロモーション/OT) コンプライアンスを守るために

国名	メッセージの種類	形式の制限と要件	登録の要件
			は、送信者 ID がカタール宛てのトランザクション/OTP メッセージのみであることに注意してください。 • この送信者 ID によって送信されるすべてのメッセージがユースケースに記述されていることの書面による確認 • 送信する予定のメッセージテンプレート

国名	メッセージの種類	形式の制限と要件	登録の要件
ロシア (RU)	トランザクションメッセージのみ	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 納税者番号またはライセンス番号 • ビジネス登録証明書 • 連絡先 E メールアドレス • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください

国名	メッセージの種類	形式の制限と要件	登録の要件
			• このユースケースがこのアカウントからロシアに送信されるすべてのメッセージに適用されることの確認 • トランザクション以外のメッセージは別の送信者 ID を使用して送信する必要があることの確認 • 月額料金 272 USD。この定期的な月額料金を承認することを確認してください: はい/いいえ • 送信する予定のメッセージテンプレート

国名	メッセージの種類	形式の制限と要件	登録の要件
サウジアラビア (SA)	各送信者 ID は、トランザクション用かプロモーション用のいずれかである必要があります。1 つの送信者 ID を両方のタイプのトラフィックに使用することはできません。OTP または 2FA トラフィックを送信する場合、送信者 ID はその目的にのみ使用する必要があります。プロモーション用の送信者 ID は、サウジアラビアのサイレントモード (DND) リストの対象となります。	• プロモーション送信者 ID の長さ: 2 ~ 8 文字、送信者 ID の先頭に「-AD」が付きます • トランザクション送信者 ID の長さ: 2 ~ 11 文字 • 送信者 ID は送信者のブランドアイデンティティを表す必要があります • 少なくとも 1 つの文字を含めます • ASCII 特殊文字 (#、@ など) は使用しないでください • 大文字と小文字、0 ~ 9 の数字を使用できます。	サウジアラビアでの送信者 ID 登録のサポートは、国際企業のみを対象としています。現在、サウジアラビアの現地企業による送信者 ID の登録はサポートしていません • 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信

国名	メッセージの種類	形式の制限と要件	登録の要件
			者 ID の関係について説明してください • 送信する予定のメッセージテンプレート。 • この送信者 ID はトランザクションコンテンツやプロモーションコンテンツに使用されますか。 • トランザクション以外のメッセージは別の送信者 ID を使用して送信する必要があることの確認 • 2FA または OTP トラフィックを送信する場合に、この送信者 ID がこの目的にのみ使用されることの確認
シンガポール (SG)	トランザクションメッセージのみ	• 最大 11 文字 • スペースは使用できません • 特殊文字なし	「シンガポールの送信者 ID 登録要件」を参照してください。

国名	メッセージの種類	形式の制限と要件	登録の要件
スリランカ (LK)	制限や特別な要件はありません	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 企業タイプ (国内/国際) • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • ユースケースの説明、メッセージの目的 • 送信する予定のメッセージテンプレート • この送信者 ID はトランザクションコンテンツやプロモーションコンテンツに使用されますか。 • トランザクション以外のメッセージは別の送信者 ID を使用して送信する必要があることの確認 • 2FA または OTP トラフィックを送信する場合に、この送信者 ID がこの目

国名	メッセージの種類	形式の制限と要件	登録の要件
			的にのみ使用されることの確認

国名	メッセージの種類	形式の制限と要件	登録の要件
タイ (TH)	制限や特別な要件はありません	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください • 送信される SMS のタイプ (トランザクション/プロモーション/OTP)

国名	メッセージの種類	形式の制限と要件	登録の要件
			• 送信する予定のメッセージテンプレート

国名	メッセージの種類	形式の制限と要件	登録の要件
トルコ (TR)	制限や特別な要件はありません	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 会社がトルコに拠点を置いている場合または海外に拠点を置いている場合 • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください • 送信される SMS のタイプ (トランザクション/プロモーション/OTP)

国名	メッセージの種類	形式の制限と要件	登録の要件
			• 送信する予定のメッセージテンプレート

国名	メッセージの種類	形式の制限と要件	登録の要件
ウクライナ (UA)	制限や特別な要件はありません	• 英数字 • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • VAT 番号 • 国内企業または国際企業 • 月間使用量の見積もり • ユースケースの説明、メッセージの目的 • 明確でない場合は、会社名と送信者 ID の関係について説明してください • 送信される SMS のタイプ (トランザクション/プロモーション/OTP)

国名	メッセージの種類	形式の制限と要件	登録の要件
			• 送信する予定のメッセージテンプレート

国名	メッセージの種類	形式の制限と要件	登録の要件
アラブ首長国連邦 (UAE)	トランザクションメッセージのみ	• 英数字 • 汎用送信者 ID は使用できません。ブランドを特定する必要があります • 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 会社名 • 会社の住所 (会社の市区町村、都道府県、郵便番号を含む) • 会社の国 • 会社の連絡先電話番号 • 会社の URL (アプリまたは会社のウェブサイトへのリンク) • 月間使用量の見積もり • 明確でない場合は、会社名と送信者 ID の関係について説明してください • ユースケースの説明、メッセージの目的 • 会社の正式なビジネス/貿易ライセンス番号または VAT 番号 • この送信者 ID によって送信され

国名	メッセージの種類	形式の制限と要件	登録の要件
			るすべてのトラフィックが、指定されたユースケースで説明されていることを書面で確認する • 送信する予定のメッセージテンプレート
ベトナム (VN)	トランザクションメッセージのみ。マーケティングやプロモーションのメッセージは許可されていません。禁止されているコンテンツには以下が含まれます。 • アダルト向けコンテンツ • 慈善サービス • Cryptocurrency • 宝くじ • モバイルギャンブル/カジノ • スポーツくじ • 投票	• 最大 11 文字 • スペースは使用できません • 特殊文字なし	• 登録する送信者 ID • ユーザーが API/サービスを呼び出す AWS リージョン • 月間使用量の見積もり • 明確でない場合は、会社名と送信者 ID の関係について説明してください • ユースケースの説明、メッセージの目的 • この送信者 ID によって送信されるすべてのトラフィックが、指定されたユースケースで説明されていることを書面で確認する

署名形式の制限と要件

中国への配信を成功させるには、メッセージコンテンツテンプレートにメッセージ署名を登録する必要があります。このメッセージ署名は、送信する各 SMS のメッセージ内容の先頭に付加する必要があります。SMS メッセージ本文の先頭に登録済みの署名を付加しないと、SMS がブロックまたはフィルタリングされる可能性があります。

- SMS 本文の先頭には角括弧で囲んで署名を付ける必要があります
- 標準テキストは角括弧で囲む必要があります
- Unicode テキストには署名を入れるにはレンズ状ブラケットを使用する必要があります - U+3010 左レンズ状ブラケット、U+3011 右レンズ状ブラケット - 例: 【注意】
- 3～11 文字である必要があります
- 中国語/英語の文字がサポートされています

フランスの送信者 ID の要件

このガイドでは、フランスの携帯電話会社がフランスに SMS テキストメッセージを送信するために必要な専用送信者 ID を作成するために必要な手順とガイドラインについて説明します。

トピック

- フランス専用の送信者 ID の設定
- 送信者 ID 命名のガイドライン

フランス専用の送信者 ID の設定

次のいずれかの方法を使用できます。Amazon SNS は、Publish API を使用して公開される SMS メッセージの送信者 ID をお客様に代わって使用します。

- Amazon SNS コンソールを使用して、公開された SMS メッセージを表示するために Amazon SNS コンソールを使用できます。詳細については、「を使用した SMS メッセージングプリファレンスの設定 AWS Management Console」を参照してください。
- Amazon SNS に SMS メッセージの公開を要求する際、AWS.SNS.SMS.SenderID メッセージ属性を使用した送信者 ID の設定に Publish API を使用できます。詳細については、「メッセージの送信 (コンソール)」を参照してください。

送信者 ID 命名のガイドライン

- 送信者 ID 名は最大 11 文字の英数字である必要があります。
- 送信者 ID 名に特殊文字またはスペースを含めることはできません。
- 送信者 ID と SMS テキストメッセージを送信する会社のブランド名には同じ名前を使用することをお勧めします。

インドの送信者 ID 登録要件

デフォルトでは、インドの受信者にメッセージを送信すると、Amazon SNS は International Long Distance Operator (ILDO) 接続を使用してこれらのメッセージを送信します。受信者が ILDO 接続で送信されたメッセージを表示すると、ランダムな数字の ID から送信されたように見えます (専用のショートコードを購入した場合を除く)。

> **Note**
>
> ローカルルートを使用したメッセージの送信料金は、[Amazon SNS 世界各国の SMS 料金] ページに表示されます。ILDO 接続を使用してメッセージを送信する料金は、ローカルルート経由でメッセージを送信する料金よりも高くなります。

SMS メッセージにアルファベットの送信者 ID を使用する場合は、ILDO ルートではなくローカルルートでメッセージを送信する必要があります。ローカルルートを使用してメッセージを送信するには、まず分散型台帳 (DLT) ポータルを介してユースケースとメッセージテンプレートをインド電気通信規制庁 (TRAI) に登録する必要があります。これらの登録要件は、インドの消費者が受信する未承諾メッセージの数を減らし、潜在的に有害なメッセージから消費者を保護するために設定されています。この登録プロセスは、Vilpower サービスを通じて Vodafone India によって管理されています。

トピック

- ステップ 1: TRAI に登録する
- ステップ 2: 送信者 ID をリクエストする
- ステップ 3: SMS メッセージを送信する
- インドの受信者に送信した SMS メッセージのトラブルシューティング

ステップ 1: TRAI に登録する

インドの受信者に SMS メッセージを送信するには、事前に組織をインド電気通信規制庁 (TRAI) に登録する必要があります。登録手続きの際に、以下の情報を入力する準備をしておいてください。

- 組織の Permanent Account Number (PAN)。
- 組織の Tax Deduction Account Number (TAN)。
- 組織の Goods and Services Tax Identification Number (GSTIN)。
- 組織の Corporate Identity Number (CIN)。
- 組織を登録する権限を与える承認書。

以下は、組織を TRAI に登録するために利用可能な分散型台帳 (DLT) 登録サイトのサンプルリストです (手数料がかかる場合があります)。登録プロセスはサイトによって異なります。サポートについては、それぞれのサポートチームにお問い合わせください。

- BSNL DLT - 登録は無料です。
- Jio TrueConnect - 登録プロセスの完了に料金を請求します。
- Smart Enterprise Solutions - 登録プロセスの完了に料金を請求します。
- Vilpower - ダウンロードして自由に変更可能なテンプレートがあります。Vilpower は、登録プロセスを完了するための手数料を請求します。

組織を TRAI に登録するには

Vilpower を使用して組織を TRAI に登録する方法は、以下の詳細を参照してください。

1. ウェブブラウザで、Vilpower のウェブサイト https://www.vilpower.in にアクセスします。
2. [Signup] を選択して、別のアカウントを作成します。登録処理中に、次の操作を行います。
   - 登録するエンティティのタイプは、[As Enterprise] を選択します。
   - テレマーケティング担当者名は、[Infobip Private Limited - ALL] を使用します。プロンプトが表示されたら、**`Infobip`** を入力し、次に、ドロップダウンリストから [Infobip Private Limited – ALL] を選択します。
   - [Enter Telemarketer ID] に、**`110200001152`** を入力します。
   - ヘッダー ID の入力を求められたら、登録する送信者 ID を入力します。

> **Note**
>
> インドでは、送信者 ID の長さは正確に 6 文字である必要があります。

- コンテンツテンプレートの入力を求められたら、受信者に送信する予定のメッセージコンテンツを入力します。送信するすべてのメッセージのテンプレートを含めます。

> **Note**
>
> Amazon Web Services では、DLT 登録プロバイダーのウェブサイトは管理していません。それぞれのウェブサイトのステップは変更される可能性があります。

ステップ 2: 送信者 ID をリクエストする

インドで送信者 ID をリクエストするには、AWS Support リクエストを提出する必要があります。「送信者 ID をリクエストする」で各ステップを実行します。リクエストで以下の必須情報を入力してください。

- 送信者が SMS メッセージを送信する予定の AWS リージョン。
- DLT 登録プロセス中に使用する会社名。
- DLT エンティティの登録が成功した後に受け取ったプリンシパルエンティティ ID (PEID)。
- 月間使用量の見積もり。
- ユースケースの説明。
- エンドユーザーオプトインフローの説明。
- エンドユーザーのオプトインが収集され、登録されていることの確認。

ステップ 3: SMS メッセージを送信する

組織を TRAI に登録したら、インドの受信者に SMS メッセージを送信できます。

1. Amazon SNS コンソールにサインインします。
2. コンソールメニューで、SMS メッセージングをサポートしているリージョンにリージョンセレクタを設定します。
3. ナビゲーションパネルで、[テキストメッセージング (SMS)] を選択します。

4. [テキストメッセージ (SMS)] ページで、[テキストメッセージの発行] を選択します。[SMS メッセージの発行] ウィンドウが開きます。
5. [メッセージタイプ] で、以下のいずれかを選択します。

   - プロモーション - マーケティングメッセージなどの非クリティカルなメッセージ。

     数値の送信者 ID を使用する場合は、このオプションを選択します。
   - [トランザクション] - 多要素認証のワンタイムパスコードなど、顧客のトランザクションをサポートするクリティカルメッセージ。

     アルファベットまたはアルファベットと数字の送信者 ID を使用する場合は、このオプションを選択します。

   このメッセージレベルの設定は、[テキストメッセージプリファレンス] ページで設定するデフォルトメッセージタイプを上書きします。

   プロモーションおよびトランザクションメッセージの料金表の詳細については、「グローバル SMS 料金表」を参照してください。
6. メッセージを送信する電話番号を [番号] に入力します。
7. [メッセージ] に、送信するメッセージを入力します。

   SMS メッセージにコンテンツを追加するときは、コンテンツが DLT 登録済みテンプレートのコンテンツと完全に一致していることを確認してください。通信事業者は、メッセージコンテンツに追加の返ってくる文字、スペース、句読点、または文の大文字と小文字が一致しない場合、SMS メッセージをブロックします。テンプレート内では、30 文字以内で変更する可能性のある文字を指定できます。
8. [発信元 ID] セクションで、[送信者 ID] に、3～11 文字のカスタム ID を入力します。

   送信者 ID は、プロモーションメッセージの場合は数値、トランザクションメッセージの場合はアルファベットまたはアルファベットと数字です。送信者 ID は、受信側デバイスにメッセージ送信者として表示されます。

   インドに登録されているプロモーション用の送信者 ID の数値については、SMS 送信リクエストの [発信元番号] パラメータとして送信者 ID を指定します。
9. インドの受信者に SMS メッセージを送信する場合は、[国固有の属性] セクションを展開し、次の必須属性を指定します。

- エンティティ ID - インドの受信者に SMS メッセージを送信するためのエンティティ ID または プリンシパルエンティティ (PE) ID。

  これは、TRAI に登録したエンティティを一意に識別する、TRAI が付与する 1～50 文字のカスタム文字列です。

- テンプレート ID - インドの受信者に SMS メッセージを送信するための、規制機関から受け取るテンプレート ID。

  これは、TRAI に登録したテンプレートを一意に識別する、TRAI が付与する 1～50 文字のカスタム文字列です。テンプレート ID は、前のステップで指定した送信者 ID とメッセージコンテンツに関連付ける必要があります。

10. [メッセージの発行] を選択します。

その他の国の受信者に SMS メッセージを送信する方法の詳細については、「携帯電話に発行する」を参照してください。

インドの受信者に送信した SMS メッセージのトラブルシューティング

以下は、通信事業者がSMSメッセージをブロックする理由と考えられるものです。

- 送信されたコンテンツに一致するテンプレートが見つからなかった。

  送信されたコンテンツ: **<#> 12345 is your OTP to verify mobile number. Your OTP is valid for 15 minutes -- ABC Pvt. Ltd.**

  一致テンプレート: なし

  問題: 登録した DLT テンプレートの最初に <#> または{#var#} を含む DLT テンプレートがない。

- 変更する可能性のある文字数が 30 文字を超えている。

  送信されたコンテンツ: **12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  一致したテンプレート: **{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC Pvt. Ltd.**

問題: 送信されたコンテンツの「ABC Company - India Private Limited」の値が、単一の `{#var#}` 文字数制限の 30 を超えている。

- メッセージ文の大文字と小文字がテンプレートの文の大文字と小文字と一致しません。

  送信されたコンテンツ: **`12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.`**

  一致したテンプレート: **`{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC PVT. LTD.`**

  問題: 一致した DLT テンプレートに付け加えられた会社名はすべて大文字で、送信されたコンテンツでは名前の一部が小文字に変更された (「ABC Pvt. Ltd.」と 「ABC PVT。Ltd.」)。

シンガポールの送信者 ID 登録要件

Amazon SNS のお客様は、シンガポール SMS 送信者 ID レジストリ (SSIR) を通じて登録された送信者 ID を使用して、シンガポールで SMS トラフィックを送信できます。SSIR は、シンガポールの情報通信メディア開発庁 (IMDA) が所有するシンガポールネットワークインフォメーションセンター (SGNIC) を通じて 2022 年 3 月に開始され、シンガポール内の携帯電話に SMS を送信する際に、組織は送信者 ID を登録できるようになりました。

登録されたシンガポールの送信者 ID を使用するには、固有エンティティ番号 (UEN) の取得後、送信者 ID を使用するためにアカウントを許可リストに登録するように Amazon にリクエストして、最後に SSIR 経由で登録プロセスを完了します。

2023-01-30 までに ID を登録しない場合、送信者 ID を使用して送信されたメッセージは、規制当局の規則に従って、ID が LIKELY-SCAM に変更されます。この日以降も、規制当局は引き続き未登録のトラフィックを独自の裁量でフィルタリングまたはブロックします。

> **Important**
>
> Amazon Pinpoint リージョンで送信者 ID をリクエストしている場合は、Amazon Pinpoint コンソールを使用して送信者 ID を登録します。Amazon Pinpoint リージョン以外のリージョンの登録プロセスを手動で完了するには、シンガポールの送信者 ID 登録を使用してください。
> シンガポールで引き続きメッセージを送信できるようにするには、2023—01-30 までに登録を完了する必要があります。

次の順序で登録手順を完了することが非常に重要です。これらの手順を順不同で実行すると、サービスに送信者 ID がブロックされる場合や送信者 ID がモバイルデバイス上で保持されない場合があります。
ステップ 1. シンガポール固有エンティティ番号 (UEN) の登録
[Step 2.](ステップ2.) Amazon Pinpoint リージョンで送信者 ID をリクエストしている場合は、Amazon Pinpoint 送信者 ID 登録手順を使用して送信者 ID を登録してください。

- アカウントが Amazon Pinpoint リージョンにない場合、シンガポールの送信者 ID の登録手順を使用して、手動で送信者 ID を登録します。
- 別の会社に代わって SMS テキストメッセージを送信する場合、会社からの認証書 (LOA) が必要です。
- AWS 送信者 ID 登録を送信した後は、承認やステータスの変更を待たないでください。すぐにステップ 3 に進みます。

ステップ 3. シンガポールネットワーク情報センター (SGNIC) への送信者 ID の登録

トピック

シンガポール固有エンティティ番号 (UEN) の登録

SSIR への登録を開始するには、まずシンガポール固有エンティティ番号 (UEN) を取得する必要があります。UEN は、会計企業規制庁 (ACRA) に事業を登録したときに受け取る固有のエンティティ番号です。詳細については、「Who Must Register with ACRA?」(ACRA に登録する必要があるのは誰

ですか?) を参照してください。処理にかかる時間は、ACRA がリクエストをどの程度容易に検証できるかによって異なります。

シンガポールの送信者 ID を Amazon Pinpoint に登録する

シンガポール固有エンティティ番号 (UEN) を登録すると、Amazon Pinpoint コンソールで送信者 ID 登録プロセスを完了できます (Amazon Pinpoint リージョンのみ)。送信者 ID を登録するときは、情報が完全かつ正確であることを確認してください。そうでないと、登録が却下される場合があります。

> **Important**
>
> Amazon Pinpoint コンソール経由で送信された情報は、登録を完了するために当社の通信事業者パートナーに渡されます。

シンガポールの送信者 ID を登録するには

アカウントが Amazon Pinpoint リージョンにある場合は、以下のステップを使用して送信者 ID を登録します。アカウントが Amazon Pinpoint リージョンにない場合は、「シンガポールの送信者 ID 登録を完了する手動登録プロセス」を参照してください。

1. AWS マネジメントコンソールにサインインして、https://console.aws.amazon.com/pinpoint/ で Amazon Pinpoint コンソールを開きます。
2. ナビゲーションペインで、[SMS and voice] (SMS と音声) の [Phone numbers] (電話番号) を選択します。
3. [Sender ID registrations] (送信者 ID 登録) タブで、[Create registration] (登録の作成) を選択します。
4. 宛先国として [Singapore] (シンガポール) を選択します。
5. [Company infomation] (会社情報) セクションで、次を入力します。
   - [Company Name] (会社名) には、UEN 登録に表示されているとおり正確に会社名を入力します。
   - [Tax ID] (納税 ID) には、ACRA から受け取った UEN 番号を入力します。
   - [Company Website] (会社のウェブサイト) に、会社のウェブサイトの URL を入力します。
   - [Address 1] (住所 1) に、本社の住所を入力します。
   - [Address 2] (住所 2) (オプション) に、必要に応じて本社の部屋番号を入力します。

- [City] (市) に、本社の市を入力します。
- [State] (州) に、本社の州を入力します。
- [Zip Code] (郵便番号) に、本社の郵便番号を入力します。
- [Country] (国) には、2 桁の ISO 国コードを入力します。

6. [Contact information] (連絡先情報) セクションで、次の情報を入力します。
   - [First Name] (名) に、連絡先となる担当者の名を入力します。
   - [Last Name] (姓) に、連絡先となる担当者の姓を入力します。
   - [Support Email] (サポート E メール) に、連絡先となる担当者の E メールアドレスを入力します。
   - [Support Phone Number] (サポート電話番号) に、連絡先となる担当者の電話番号を入力します。
7. [Sender ID Information] (送信者 ID 情報) に、以下を入力します。
   - [Sender ID] (送信者 ID) に、メッセージに表示する送信者 ID を入力します。
   - [Registering on behalf of another brand/entity?] (別のブランド/エンティティを代表して登録しますか?) については、[Yes] (はい) の場合、[True] を選択します。メッセージを送信するエンドユーザーでない場合、ユーザーは他のブランド/エンティティの「代表者」と見なされます。
   - [Letter of authorization image] (認可書の画像) (オプション) については、[Registering on behalf of another brand/entity?] (別のブランド/エンティティを代表して登録しますか?) のボックスをオフにした場合、完全な認可書 (LOA) の画像をアップロードします。サポートされるファイルタイプは PNG で、ファイルの最大サイズは 400 KB です。LOA のテンプレートをダウンロードしておくと便利です。
   - [Sender ID] (送信者 ID) 接続 — オプションで、リクエストされた SenderID と会社名間の接続に関する詳細を追加できます。
8. [Messaging Use Case] (メッセージングユースケース) で、以下の操作を行います。
   - [Monthly SMS Volume] (毎月の SMS ボリューム) で、毎月送信する SMS メッセージの数を選択します。
   - [Use Case Category] (ユースケースカテゴリ) で、番号に対するユースケースタイプのいずれかを以下から選択します。
     - [Two-factor authentication] (2 要素認証) — これを使用して 2 要素認証コードを送信します。
     - [One-time passwords] (ワンタイムパスワード) — これを使用してユーザーにワンタイムパスワードを送信します。
     - [Notifications] (通知) — これはユーザーに重要な通知を送信する場合にのみ使用します。

- [Polling and surveys] (ポーリングおよびアンケート) — これを使用してユーザーの好みを調査します。
- [Info on demand] (オンデマンド情報) — ユーザーから送信されたリクエストに応じてユーザーにメッセージを送信します。
- [Promotions and Marketing] (プロモーションおよびマーケティング) — これはユーザーにマーケティングメッセージを送信する場合にのみ使用します。
- [Other] (その他) — ユースケースが他のどのカテゴリにも該当しない場合に、これを使用します。このオプションの [Use Case Details] (ユースケースの詳細) は必ず入力してください。

- [Use Case Details] (ユースケースの詳細) を完了 - オプションで、選択した [Use Case Category] (ユースケースカテゴリ) に追加のコンテキストを指定します。

9. [Messaging Samples] (メッセージングのサンプル) セクションで、次の操作を行います。
   - [Message Sample 1] (メッセージサンプル 1) に、エンドユーザーに送信する SMS メッセージ本文のサンプルメッセージを入力します。
   - [Message Sample 2] (メッセージサンプル 2) — オプションおよび [Message Sample 3] (メッセージサンプル 3) – オプションには、必要に応じて、送信される SMS メッセージ本文の追加のサンプルメッセージを入力します。
   - 各 [Message Sample] (メッセージサンプル) テキストボックスの最大文字数は 306 文字です。
10. 完了したら、[Submit registration] (登録の送信) を選択します。

> **Important**
>
> 登録のステータスは、シンガポールの送信者 ID 登録ステータス の次の指示に従って確認できます。
> 送信者 ID 登録を送信した後は、承認やステータスの変更を待たないでください。すぐにシンガポールネットワーク情報センター (SGNIC) への送信者 ID の登録 に移動します。

シンガポールの送信者 ID 登録を完了する手動登録プロセス

アカウントが Amazon Pinpoint リージョンにない場合は、以下のステップを使用して送信者 ID を登録します。アカウントが Amazon Pinpoint リージョンにある場合は、「シンガポールの送信者 ID を Amazon Pinpoint に登録する」を参照してください。

1. Singapore_Sender_ID_Registration_LOA_Template.zip をダウンロードして、必要な情報を入力します。

2. AWS サポートでケースを作成します。
3. [Open support cases] タブで、[ケースの作成] を選択します。
4. [Looking for service limit increases] (サービスの制限を引き上げる) を選択し、制限タイプで [SNS Text Messaging] (SNS テキストメッセージ) を選択します。
5. [リソースタイプ] で、[Sender ID Registration] を選択します。
6. LOA 文書を添付してリクエストを送信します。

シンガポールネットワーク情報センター (SGNIC) への送信者 ID の登録

> **Warning**
>
> これらの手順を順不同で実行すると、サービスに送信者 ID がブロックされる場合や送信者 ID がモバイルデバイス上で保持されない場合があります。

1. まず、AWS を使用して、アカウントのシンガポール (SG) 送信者 ID を登録する必要があります (Amazon Pinpoint コンソール、または Amazon Pinpoint 以外のリージョンの場合は手動登録)。この手順が完了したら、次のステップに進むことができます。
2. SGNIC SMS 送信者 ID 登録のプロセスを使用して送信者 ID を登録するには、SGNIC と連携します。
   - プロセスを完了するとき、以下のすべてを参加アグリゲーターとしてリストアップしてください。
     - AMCS SG Private Limited (Amazon Media Communications Services)
     - Nexmo PTE LTD
     - Sinch Singapore PTE LTD
     - Telesign Singapore PTE LTD
     - Twilio Singapore PTD LTD

> **Note**
>
> 送信者 ID を使用するために必要なそれぞれの AWS アカウントから送信者 ID 登録を送信する必要があります。

シンガポールの送信者 ID 登録ステータス

シンガポールの送信者 ID を Amazon SNS に登録すると、登録は次の 5 つの異なるステータスのいずれかになります。

- [Created] (作成済み) — 登録は作成されましたが、送信されていません。
- [Submitted] (送信済み) — 登録が送信され、検証中です。
- レビュー — 登録が承認され、レビュー中です。1 ~ 3 週間かかる場合があり、レビューが完了するまでにさらに時間がかかることもあります。
- [Complete] (完了) – 登録は承認済みであり、送信者 ID の使用を開始できます。
- [Requires Updates] (更新が必要) — 登録を修正して再送信する必要があります。詳細については、「シンガポールの送信者 ID 登録の編集」を参照してください。更新が必要なフィールドには、警告アイコンと問題の簡単な説明が表示されます。

Amazon Pinpoint リージョン以外のすべてのリージョンの場合、[AWS Support] (AWS サポート) は登録時に確認メールを送信するか、[AWS Support] (AWS サポート) でケースを作成します。

- [Open support cases] タブで、[ケースの作成] を選択します。
- [Service Limit increase] を選択します。
- [Resource Type] (リソースタイプ) で、[Sender ID Registration] (送信者 ID 登録) を選択し、制限には [General Inquiry] (一般的なお問い合わせ) を選択します。

登録のステータスを確認する

1. AWS マネジメントコンソールにサインインして、https://console.aws.amazon.com/pinpoint/ で Amazon Pinpoint コンソールを開きます。
2. ナビゲーションペインで、[SMS and voice] (SMS と音声) の [Phone numbers] (電話番号) を選択します。
3. [Sender ID registrations] (送信者 ID 登録) タブで、[SenderID] を選択します。
4. 各 SenderID の登録ステータスを確認できます。

シンガポールの送信者 ID 登録の編集

Amazon Pinpoint への登録を送信した後で、登録に問題がある場合、[registration status] (登録ステータス) が [Requires Updates] (更新が必要) と表示されます。この状態では、登録フォームは編集可能です。更新が必要なフィールドには、警告アイコンと問題の簡単な説明が表示されます。

送信者 ID を編集するには

1. Amazon Pinpoint コンソール (https://console.aws.amazon.com/pinpoint/) を開きます。
2. ナビゲーションペインで、[SMS and voice] (SMS と音声) の [Phone numbers] (電話番号) を選択します。
3. [SenderID Registration] (SenderID の登録) タブで、編集する番号を選択し、[Registration ID] (登録 ID) を選択します。
4. [Update Registration] (登録の更新) を選択してフォームを編集し、警告アイコンのあるフィールドを修正します。
5. 別のブランド/エンティティを代表して登録する場合は、以前に提出した認可書の画像のファイルを再アップロードする必要があります (オプション)。
6.

> **Important**
>
> すべてのフィールドを再チェックして、正しいことを確認します。

7. 完了したら、[Submit registration] (登録の送信) を選択します。

シンガポールの送信者 ID 登録の編集

シンガポールの送信者 ID の登録を続行しない場合は、登録を削除できます。登録は、ステータスが [Created] (作成済み) または [Requires Updates] (更新が必要) の場合にのみ削除できます。

登録を削除するには

1. Amazon Pinpoint コンソール (https://console.aws.amazon.com/pinpoint/) を開きます。
2. ナビゲーションペインで、[SMS and voice] (SMS と音声) の [Phone numbers] (電話番号) を選択します。
3. [Sender ID] (送信者 ID) タブで、削除する登録 ID を選択し、[Delete Registration] (登録の削除) を選択します。

シンガポール登録の問題

シンガポールの送信者 ID が Amazon Pinpoint で承認されない場合は、拒否の理由を説明するメッセージが表示されます。この拒否について、ベストプラクティスで回答されていない質問がある場合は、サポートチームにリクエストを送信できます。

拒否されたシンガポールの送信者 ID に関する情報のリクエストを提出するには

1. Amazon Pinpoint コンソール (https://console.aws.amazon.com/pinpoint/) を開きます。
2. [Support] を選択し、次に [Support Center] を選択します。
3. [Support] ページで、[Create case] を選択します。
4. [case type] で、[Service limit increase] を選択します。
5. [Limit Type] で、[Pinpoint SMS] を選択します。
6. [Requests] セクションで、以下の操作を行います。
   - [Resource Type] (リソースタイプ) で、[Sender ID Registration] (送信者 ID 登録) を選択します。
   - [Limit] (制限) で、[Registration Rejection Query] (登録拒否クエリ) を選択します。
7. [Use case description] (ユースケースの説明) には、拒否されたシンガポールの送信者 ID と提供された拒否理由を入力します。
8. [連絡先のオプション] の [優先する連絡先の言語] で、AWS サポートチームとの連絡に使用する言語を選択します。
9. [連絡方法] では、AWS サポートチームとやり取りする方法を指定します。
10. [送信] を選択します。

AWS サポートチームは、AWS サポートケースで 送信者 ID 登録が拒否された理由についての情報を提供します。

シンガポールの送信者 ID 登録に関するよくある質問

シンガポールの送信者 ID 番号を Amazon Pinpoint に登録するプロセスに関するよくある質問。

自分は、現在、シンガポールの送信者 ID を持っていますか?

シンガポールの送信者 ID を所有しているかどうかを確認するには

1. Amazon Pinpoint コンソール (https://console.aws.amazon.com/pinpoint/) を開きます。
2. ナビゲーションペインで、[SMS and voice] (SMS と音声) の [Phone numbers] (電話番号) を選択します。
3. [SenderID Registration] (SenderID の登録) タブで、表示する 送信者 ID を選択し、[Registration ID] (登録 ID) を選択します。

登録にはどのくらい時間がかかりますか?

レビューには通常 1~3 週間かかりますが、政府機関に情報を確認するまでに最大で 5 週間以上かかることもあります。

固有エンティティ番号 (UEN) とは何ですか? どうすれば取得できますか?

UEN は、会計企業規制庁 (ACRA) が発行するシンガポールのビジネス ID です。シンガポールの地元の会社や企業は、ACRA を通じて申請することで UEN を取得できます。登録と標準の法人設立手続きを通過すると、発行されます。Bizfile 経由で ACRA により UEN を申請できます。

自分は、シンガポールの送信者 ID を登録する必要がありますか?

はい。現在、2023-01-30 までにシンガポールの送信者 ID を登録していない場合、送信者 ID を使用して送信されたメッセージの ID は LIKELY-SCAM に変更されます。

シンガポールの送信者 ID を Amazon Pinpoint に登録するにはどうすればいいですか?

「シンガポールの送信者 ID を Amazon Pinpoint に登録する」の指示に従って、送信者 ID を登録します。

自分のシンガポールの送信者 ID の登録ステータスはどうなっていて、そのステータスは何を意味しますか?

「シンガポールの送信者 ID 登録ステータス」の指示に従って登録とステータスを確認します。

どのような情報を提供すればよいですか?

会社の住所、担当者の連絡先、ユースケースを指定する必要があります。必要な情報は、「シンガポールの送信者 ID を Amazon Pinpoint に登録する」で確認できます。

シンガポールの送信者 ID の登録が却下された場合はどうなりますか?

登録が却下されると、そのステータスが [Requires Updates] (更新が必要) に変更され、「シンガポールの送信者 ID 登録の編集」の指示に従って更新することができます。

どのようなアクセス許可が必要ですか?

Amazon Pinpoint コンソールにアクセスするために使用する IAM ユーザー/ロールは、*"sms-voice:*"* アクセス許可で有効にする必要があります。

送信元番号

発信元番号は、SMS メッセージの送信者の電話番号を識別する数値文字列です。送信元番号を使用して SMS メッセージを送信すると、受信者のデバイスには送信元番号が送信者の電話番号として表示されます。ユースケースごとに異なる送信元番号を指定できます。

> **Tip**
>
> お客様の AWS アカウントの既存の送信元番号のリストを表示するには、[Amazon SNS コンソール] のナビゲーションペインで、[送信元番号] を選択します。

送信元番号のサポートは、送信元番号ではなく送信者 ID の使用が現地法により義務付けられている国では利用できません。

トピック

- 10DLC
- 通話料無料の番号
- ショートコード
- パーソントゥーパーソン (P2P) ロングコード
- 米国製品番号の比較

10DLC

米国の通信事業者は、ローカルの未登録のロングコードで、Application-to-Person (A2P) SMS メッセージを使用したサポートを終了しています。大量の A2P SMS メッセージングの代わりに、米国の通信事業者は、10 桁のロングコード (10DLC) と呼ばれる新しいタイプのロングコードを提供しています。

> **Important**
>
> 2023 年 1 月 26 日から、Amazon SNS の SMS ベンダーは、米国の通信事業者から寄せられた SMS スパムの懸念に対処するため、10DLC キャンペーンに新しい手動のレビュープロセスを導入しました。米国で SMS を送信する場合、10DLC の代わりにショートコードと通話料無料番号を利用できます。
> 現在、当社の SMS ベンダーは、10DLC キャンペーンのレビューにかかる時間に関して、サービスレベルの目標を提示していません。10DLC キャンペーンに数字を関連付けると、レ

ビューがトリガーされます。Amazon SNS から以前にお知らせした 14 日間の予定時間よりも、レビューに時間がかかっています。
Amazon SNS は、以下のことを確実に行うために、SMS ベンダーと日々協力しています。

- ベンダーは、保留中の 10DLC キャンペーンのレビューをできるだけ早く完了させます。
- ベンダーはバックログの AWS リクエストを優先します。

10DLC キャンペーンの状況は、10DLC キャンペーンの指示に従って確認できます。10DLC キャンペーンの承認に追加情報が必要な場合は、AWS サポートチームから通知されます。米国の通話料無料番号は、10DLC 番号を取得するよりも早く登録できる場合があります。米国の通話料無料番号と登録プロセスの詳細については、「通話料無料番号の登録の要件とプロセス」を参照してください。

10DLC とは

10DLC は、10 桁の電話番号形式を使用した大量の A2P SMS メッセージングをサポートするために、通信事業者に登録されているロングコードの一種です。Amazon SNS は、SMS 製品としてローカルのロングコードの提供を終了し、代わりに 10DLC を提供しています。ショートコードと通話料無料の番号のみの使用であれば、10DLC の影響はありません。

10DLC とは、米国でのみ使用される 10 桁の電話番号です。10DLC から受信者に送信されるメッセージには、送信者として 10 桁の数字が表示されます。通話料無料の番号とは異なり、10DLC は従来のメッセージングとプロモーションメッセージングをサポートしており、米国の市外局番を含めることができます。

既存のローカルロングコードがある場合は、ローカルロングコードを 10DLC で有効にするよう要求できます。これを行うには、10DLC 登録プロセスを完了し、サポートチケットを送信します。10DLC のロングコードの有効化に伴って問題が発生した場合は、Amazon Pinpoint (Amazon SNS ではありません) コンソールから新しい 10DLC をリクエストするよう通知、指示されます。ロングコードを変換するためのサポートチケットの提出方法の詳細については、「ロングコードを 10DLC キャンペーンに関連付ける」を参照してください。

10DLC 番号を使用するには、まず会社を登録し、Amazon Pinpoint (Amazon SNS ではありません) コンソールを使用して 10DLC キャンペーンを作成します。AWS は、この情報をキャンペーンレジストリと共有します。キャンペーンレジストリは、この情報に基づいて登録を承認または拒否するサードパーティーです。場合によっては、登録は直ちに行われます。例えば、以前にキャンペーンレジストリに登録したことがある場合は、すでに情報が登録されている可能性があります。ただし、

キャンペーンによっては、承認に 1 週間以上かかる場合があります。会社と 10DLC キャンペーンが承認されたら、10DLC 番号を購入してキャンペーンに関連付けることができます。10DLC のリクエストには、承認までに最大 1 週間かかる場合があります。複数の 10DLC を 1 つのキャンペーンに関連付けることはできますが、複数のキャンペーンで同じ 10DLC を使用することはできません。作成するキャンペーンごとに、固有の 10DLC が必要です。

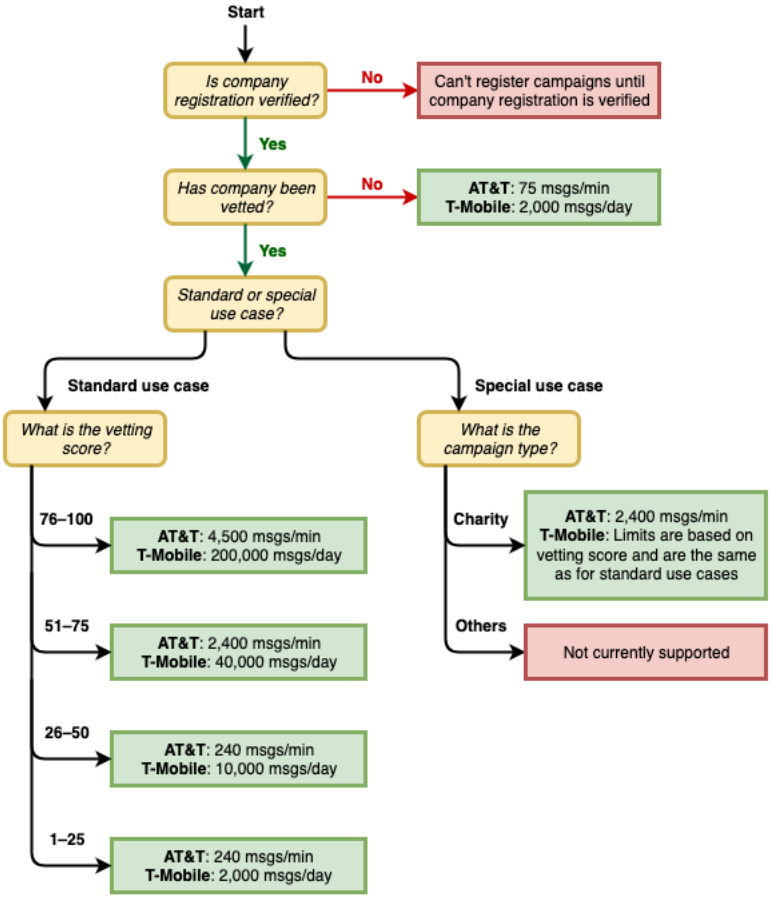
10DLC の容量

10DLC 電話番号の容量は、受信者の通信事業者によって異なります。AT&T では、キャンペーンごとに毎分送信できるメッセージパーツの数に上限があります。T-Mobile では、各社ごとに 1 日に送信可能なメッセージ数の上限が定められており、1 分間に送信可能なメッセージパーツの数は制限されていない。Verizonはスループット制限を公表していませんが、10DLC にはスパム、迷惑メッセージ、罵詈雑言などを除去するためのフィルタリングシステムを使用しており、実際のメッセージのスループットはそれほど重視されていません。

未審査の企業に関連する新規 10DLC キャンペーンでは、AT&T を利用する受信者には 1 分間に 75 通、T-Mobile を利用する受信者には 1 日 2,000 通のメッセージを送信することができます。会社の上限は、10DLC のすべてのキャンペーンで共有されます。例えば、1 社 2 キャンペーンを登録した場合、T-Mobile のお客様への 1 日 2,000 通のメッセージの割り当ては、それらのキャンペーンで共有されます。同様に、同じ会社を複数の AWS アカウントに登録した場合、1 日の割り当てはそれらのアカウントで共有されます。

スループットニーズがこの制限を超える場合は、会社登録の審査を依頼することができます。会社登録を審査する際、サードパーティーである審査機関が企業情報を分析します。審査機関は、10DLC キャンペーンの機能を決定する審査スコアを提供します。審査サービスには一回限りがあります。詳細については、「Amazon SNS の 10DLC 登録を審査する」を参照してください。

実際のスループットレートは、貴社の審査の有無、キャンペーンの種類、審査スコアなど、さまざまな要因によって変化します。以下のフローチャートは、さまざまな状況下でのスループットレートを示しています。



10DLC のスループットレートは、米国の通信事業者がキャンペーンレジストリと協力して決定しています。Amazon SNS も他の SMS 送信サービスも、このレート以上に 10DLC のスループットを上げることはできません。米国のすべての通信事業者において高いスループットレート、高い到達率が必要な場合は、ショートコードの利用をお勧めします。ショートコードの取得については、「Amazon SNS による SMS メッセージングの専用ショートコードをリクエストする」を参照してください。

10DLC の使用開始

Amazon Pinpoint (Amazon SNS ではありません) コンソールを使用して、10DLC をリクエストします。10DLC キャンペーンに対して 10DLC をセットアップするには、次のステップに従います。

1. 会社を登録します。

   10DLC をリクエストする前に、キャンペーンレジストリに登録する必要があります。詳しくは、「会社を登録する」を参照してください。キャンペーンレジストリに詳細情報が必要でない限り、通常、登録は瞬時に行われます。お客様の会社の登録には 1 回限りの登録料がかかり、これ

は登録ページに表示されます。この 1 回払いの料金は、キャンペーンおよび 10DLC の月額料金とは別に支払うものです。

> **Note**
>
> Amazon SNS SMS メッセージングは、Amazon Pinpoint が現在サポートされていないリージョンで使用できます。これには、次のように 2 つの異なるケースがあります。
>
> a. 商用クラウドアカウントを使用している場合は、米国東部 (バージニア北部) リージョンで Amazon Pinpoint コンソールを開き、10DLC の会社とキャンペーンを登録する必要があります。10DLC 番号はリクエストしないでください。
>
> b. 代わりに、AWS Service Quotas コンソールを使用して、サービス制限の増加ケースを作成するとともに、そのリージョンの 10DLC 番号をリクエストします。Amazon Pinpoint を利用できるリージョンについての詳細は、「AWS 全般のリファレンス」の「Amazon Pinpoint エンドポイントとクォータ」を参照してください。
>
> c. AWS GovCloud (US) アカウントを使用している場合は、米国西部リージョンで Amazon Pinpoint コンソールを開き、10DLC の会社とキャンペーンを登録します。10DLC 番号はリクエストしないでください。代わりに、AWSService Quotas コンソールを使用して、そのリージョンの 10DLC 番号を要求しながら、サービス制限の増加ケースを作成します。Amazon Pinpoint を利用できるリージョンについての詳細は、「AWS 全般のリファレンス」の「Amazon Pinpoint エンドポイントとクォータ」を参照してください。

2. (オプション、ただし推奨) 審査申請

   会社登録が成功すれば、少量で複合用途な 10DLC キャンペーンを開始することができます。これらのキャンペーンは、AT&T を利用している受信者には 1 分間に 75 通、T-Mobile を利用している受信者には登録企業が 1 日 2,000 通を送信することができます。ユースケースでこの値を超えるスループットレートが必要な場合は、会社登録の審査を申請することができます。会社登録を審査することで、企業やキャンペーンのスループットレートを高めることができますが、その効果を保証するものではありません。審査の詳細については、「Amazon SNS の 10DLC 登録を審査する」を参照してください。

3. キャンペーンを登録します。

   会社を登録したら、10DLC キャンペーンを作成し、登録済みの会社に関連付けます。このキャンペーンは、承認のためキャンペーン登録に送信されます。キャンペーンレジストリに詳細情報が必要な場合を除き、ほとんどの場合、10DLC キャンペーンの承認は瞬時に行われます。詳細については、「10DLC キャンペーンを登録する」を参照してください。

4. 10DLC 番号をリクエストします。

   10DLC キャンペーンが承認されたら、10DLC をリクエストし、その番号を承認済みキャンペーンに関連付けることができます。10DLC キャンペーンでは、承認された番号のみを使用できます。「Amazon SNS で SMS メッセージング用の 10DLC 番号、通話料無料番号、P2P ロングコードをリクエストする」を参照してください

10DLC 登録と月額料金

会社登録や10DLC キャンペーンなど、10DLC の使用に関連する登録料と月額料金があります。これらは、AWS が請求する他の月額料金とは別個のものです。詳細については、「Amazon SNS 世界各地の SMS 料金」ページを参照してください。

会社を登録する

10DLC をリクエストする前に、会社をキャンペーンレジストリに登録する必要があります。

> **Note**
>
> Amazon SNS SMS メッセージングは、Amazon Pinpoint が現在サポートされていないリージョンで使用できます。このような場合は、Amazon Pinpoint コンソールを米国東部 (バージニア北部) リージョンで開き、10DLC の会社とキャンペーンを登録しますが、10DLC 番号はリクエストしないでください。代わりに、AWSService Quotas コンソールを使用して、そのリージョンの 10DLC 番号を要求しながら、サービス制限の増加ケースを作成します。Amazon Pinpoint を利用できるリージョンについての詳細は、「AWS 全般のリファレンス」の「Amazon Pinpoint エンドポイントとクォータ」を参照してください。

10DLC 会社登録ステータス

企業やブランドを登録すると、未確認と確認済みの 2 つのステータスのうち、どちらかが返されます。会社登録のステータスが次の場合未確認の場合、登録に問題があったことを意味します。例えば、指定した登録会社名が、指定した納税者番号に関連付けられている会社の登録名と完全に一致しない場合があります。会社登録の詳細に問題が見つかった場合は、修正できます。会社登録の詳細の変更の詳細については、「登録会社の編集または削除」を参照してください。

会社登録のステータスが次の場合検証済みで入力した登録の詳細が正確になり、10DLC キャンペーンの作成を開始できます。

会社またはブランドを登録する

会社登録は 1 回のみ必要です。登録後は、会社や連絡先の情報を編集することができます。登録された会社を削除するには、AWS サポートでケースを作成します。会社の詳細の編集や削除については、「登録会社の編集または削除」を参照してください。

会社を登録するには

1. AWS Management Console にサインインして、Amazon Pinpoint コンソール (https://console.aws.amazon.com/pinpoint/) を開きます。
2. ナビゲーションペインで、[SMS and voice] の [Phone numbers] を選択します。
3. [10DLC campaigns] タブで、[Register company] を選択します。

> **Note**
>
> [Register your company] ページでは、登録料金が表示されます。これは、会社登録に関連する 1 回払いの料金です。この費用は、他の月額費用や手数料とは別にかかります。会社登録時、または既存の会社登録の内容を変更する際に請求されます。

4. [Company info] セクションで、次の操作を行います。

   - 法人名には、会社が登記されている名称を入力します。入力する名前は、提供する Tax ID に関連付けられた会社名と完全に一致する必要があります。

> **Important**
>
> 会社名は必ず正確に入力してください。一度送信された情報は変更できません。情報に誤りがあったり、不完全な場合、登録が遅れたり、拒否されることがあります。

   - [What type of legal form is this organization] については、お客様の会社に最も適した選択肢をお選びください。

> **Note**
>
> 米国政府そして非営利団体オプションは、米国に拠点を置く組織の登録にのみ使用できます。組織が米国以外の国に拠点を置いている場合、組織の実際の法的形態にかかわらず、非公開営利目的 組織として登録する必要があります。

- 前のステップで利益のために公開を選択した場合は、その会社の株式記号と上場している証券取引所を入力します。
- [Country of registration] は、会社が登録されている国を選択します。
- ビジネスの別名 (DBA) またはブランド名については、お客様の他のビジネス名称を入力してください。
- Tax ID には、会社の Tax ID を入力してください。ID は、会社が登録されている国によって異なります。
  - IRS 雇用者識別番号 (EIN) を持つ米国または米国以外の事業体を登録する場合は、9 桁の EIN を入力してください。法的な会社名、EIN、および住所はすべて、IRS に登録されている会社情報と一致する必要があります。
  - カナダ法人を登録する場合は、連邦法人番号または州法人番号を入力してください。CRA から提供されたビジネス番号 (BN) は入力しないでください。法的な会社名、法人番号、および住所は、すべてカナダ法人に登録されている会社情報と一致している必要があります。
  - 別の国に拠点を置く企業を登録する場合は、その国の主要な納税者番号を入力してください。多くの国では、VAT 番号の数字部分です。
- [Vertical] には、登録する会社に最も適したカテゴリーを選択してください。

5. [Contact info] セクションで、次の操作を行います。

- 住所 / ストリートには、会社の実際の住所を入力します。
- 市町村には、所在地のある市町村名を入力します。
- 都道府県には、所在地のある都道府県名を入力します。
- 郵便番号には、所在地の郵便番号を入力します。
- 会社のウェブサイトには、会社のウェブサイトの URL をフルパスで入力します。アドレスの先頭に「http://」または「https://」を含めます。
- E メールアドレスには、使用する E メールアドレスを入力します。
- 電話番号には、国番号と電話番号をハイフンなしで入力します。

> **Note**
>
> キャンペーンレジストリでは、登録情報を確認する場合に備えて、担当者の連絡先の E メールアドレスと電話番号をお伺いしています。

6. 完了したら、[作成] を選択します。キャンペーンレジストリに法人登記を送信します。ほとんどの場合、登録はすぐに受理され、ステータスが表示されます。

会社登録のステータスが検証済みの場合、少量で複合用途な 10DLC キャンペーンを開始することができます。このタイプのキャンペーンを利用すると、AT&T を利用している受信者には 1 分間に最大 75 通、T-Mobile を利用している受信者には登録企業が 1 日あたり 2,000 通のメッセージを送信することが可能です。また、Verizon や US Cellular など、米国の他の通信事業者を利用している受信者にもメッセージを送ることができます。これらの通信事業者は、スループット制限を厳しく課していませんが、10DLC メッセージにスパムや不正使用の兆候がないか厳しく監視しています。

ユースケースでこの値を超えるスループットレートが必要な場合は、会社登録の追加審査を申請することができます。ブランド登録の確認の詳細については、「Amazon SNS の 10DLC 登録を審査する」を参照してください。

会社登録のステータスが未確認の場合、提供された情報に問題があったことになります。入力された情報を確認し、すべての項目に正しい情報が含まれていることを確認します。Amazon Pinpoint コンソールで、会社登録の一部を変更することができます。会社登録の詳細の変更の詳細については、「10DLC の会社登録の編集」を参照してください。

Amazon SNS の 10DLC 登録を審査する

会社登録が成功し、よりスループットの高いキャンペーンを登録したい場合は、会社登録の審査が必要です。

登録時の審査では、サードパーティーが登録された企業情報を分析し、審査スコアを返します。高い審査スコアは、10DLC 企業やそれに関連するキャンペーンのスループットレートの向上につながります。ただし、審査はスループットを上げることを保証するものではありません。

審査スコアは遡及して適用されません。つまり、すでに 10DLC キャンペーンを作成し、後で会社登録の審査を行った場合、審査スコアは既存のキャンペーンには自動的に適用されません。このため、10DLC のキャンペーンを行う際には、事前に企業やブランドを吟味する必要があります。

> **Note**
>
> 企業やブランドの審査には、40 ドル (返金不可) の手数料がかかります。

会社登録を査定するには

1. AWS Management Console にサインインして、Amazon Pinpoint コンソール (https://console.aws.amazon.com/pinpoint/) を開きます。
2. ナビゲーションペインで、[SMS and voice] (SMS と音声) の [Phone numbers] (電話番号) を選択します。
3. [10DLC campaigns](10DLC キャンペーン) タブで、審査する [10DLC company](10DLC 会社) を選択します。
4. [company details] ページで、ページ下部の[Apply for vetting] を選択します。
5. [Apply for additional vetting] ウィンドウで、[Submit] を選択します。

   米国に拠点を置く企業の場合、審査は通常 1 分程度で完了します。米国以外の国の企業の場合、その国のデータがどれだけ容易に入手できるかにもよりますが、審査に長い時間がかかる可能性があります。

審査リクエスト送信後、[company details] ページに戻ります。[Company vetting results] セクションには、審査リクエストの状況および結果が表示されます。審査が終了すると、この表では [Score] 欄に審査結果が表示されます。10DLC のスループット能力は、審査スコアで決まります。スループットは、作成するキャンペーンの種類によって異なります。混在型やマーケティング関連の 10DLC キャンペーンを作成する場合、高いスループット率を実現するためには、他のキャンペーンタイプよりも高い審査スコアが必要です。10DLC 電話番号の機能の詳細については、「10DLC の容量」を参照してください。

審査完了後、会社登録内容に変更があった場合は、再度審査を依頼することができます。会社登記のバーティカルを変更しただけでは、バーティカルスコアは変わりません。バーティカル以外の内容を変更した場合、審査結果が変わる可能性があります。いずれの場合も、再度、審査料がかかります。

登録会社の編集または削除

Amazon Pinpoint コンソールで直接、10DLC 登録情報の一部を編集することができます。また、AWS サポートセンターでケースを作成することで、10DLC 会社登録を削除できます。

10DLC の会社登録の編集

会社の 10DLC 登録手続きが完了すると、登録内容を編集することができます。

会社登録内容を編集した後にエラーメッセージが表示された場合は、登録内容に問題がある可能性があります。 AWS サポートにチケットを開いて、詳細をリクエストできます。

会社の登録を編集するには

1. https://console.aws.amazon.com/sms-voice/ で AWS SMS コンソールを開きます。
2. 「Amazon Pinpoint SMS ユーザーガイド」の「登録の編集」の手順に従います。 Amazon Pinpoint

10DLC の会社登録の削除

会社登録を削除するには

1. https://console.aws.amazon.com/sms-voice/ で AWS SMS コンソールを開きます。
2. 「Amazon Pinpoint SMS ユーザーガイド」の「登録を削除する」の手順に従います。 Amazon Pinpoint

10DLC キャンペーンを登録する

10DLC キャンペーンを登録する際には、ユースケースの説明と、使用する予定のメッセージテンプレートを提供します。10DLC のキャンペーンを作成および登録する前に、まず法人登記が必要です。会社の登録については、「会社を登録する」を参照してください。

> **Note**
>
> 会社登録後、Amazon Pinpointは登録のステータスを 2 つのうち 1 つを表示します。確認済みまたは 未確認の 2 つのステータスが表示されます。10DLC のキャンペーン登録は、会社登録のステータスが検証済みである場合のみ完了します。少量で複合用途なキャンペーンを作成できます。
> ステータスが未確認の場合、通常、会社登録時に提供したデータの一部が不正確であることを意味します。このステータスの間は、10DLC キャンペーンを作成することはできません。会社登録の問題を解決しようとするために、会社登録を修正することができます。10DLC の会社登録の変更については、「登録会社の編集または削除」を参照してください。

このページでは、まず 10DLC キャンペーンを作成する会社の詳細を入力し、次にキャンペーンのユースケースの詳細を入力します。このページの情報は、承認のためにキャンペーンレジストリに提供されます。

このセクションでは、10DLC キャンペーンを作成する会社を選択し、さらに詳細情報を入力します。

10DLC キャンペーンの登録

1. AWS Management Console にサインインして、Amazon Pinpoint コンソール (https://console.aws.amazon.com/pinpoint/) を開きます。
2. [SMS and voice] の [Phone numbers] を選択します。
3. [10DLC campaigns] タブで、[Create a 10DLC campaign] を選択します。
4. [Create a 10DLC campaign] ページの [Campaign info] セクションで、以下を実行します。

    a. Company name で、このキャンペーンを作成する会社を選択します。会社をまだ登録していない場合は、先に登録する必要があります。会社の登録については、「会社を登録する」を参照してください。

    b. 10DLC キャンペーン名にキャンペーンの名前を入力します。

    c. [Vertical] で、会社に最もよく当てはまるオプションを選択します。

    d. ヘルプメッセージには、お客様が 10DLC の電話番号にキーワード「HELP」を送信した際に受け取るメッセージを入力してください。

    e. ストップメッセージには、お客様が 10DLC の電話番号にキーワード「STOP」を送信した際に受け取るメッセージを入力してください。

> **Tip**
>
> 「HELP」というキーワードを添えてメッセージに返信することで、お客様が受信したメッセージの詳細を知ることができます。「STOP」に返信することで、お客様はメッセージの受信を拒否することもできます。この両方のキーワードに対するレスポンスを、米国の携帯電話通信事業者に提供する必要があります。
> 以下は、米国の携帯電話通信事業者の要件に準拠している HELP レスポンスの例です。
> **`ExampleCorp Account Alerts: For help call 1-888-555-0142 or go to example.com. Msg&data rates may apply. Text STOP to cancel.`**
> 準拠している STOP レスポンスの例を以下に示します。

You are unsubscribed from ExampleCorp Account Alerts. No more messages will be sent. Reply HELP for help or call 1-888-555-0142.
これらのキーワードに対するレスポンスは 160 文字以下である必要があります。

5. [Campaign use case] セクションで、以下を実行します。

   a. [Use case type] で、チャリティ関連のユースケースの場合は、[Special] を選択します。それ以外の場合は、[Standard] を選択します。

   b. [Use case] で、あらかじめ設定されたユースケースのリストから、キャンペーンに最も近いユースケースを選択します。各ユースケースの月額料金は、ユースケース名の横に表示されます。

   > **Note**
   >
   > 10DLC キャンペーンの登録にかかる月額料金は、各ユースケースタイプの横に表示されます。10DLC のキャンペーンのほとんどは、月額使用料が同じです。少量で複合用途な登録料は、他のユースケースタイプよりも低くなります。ただし、少量で複合用途なキャンペーンでは、他のキャンペーンタイプよりも低いスループットレートがサポートされます。

   c. SMS メッセージを少なくとも 1 つ入力します。これは、顧客に送信する予定のサンプルメッセージです。この 10DLC キャンペーンで、複数のメッセージテンプレートを使用する予定がある場合は、それらも含めてください。

   > **Important**
   >
   > サンプルメッセージにプレースホルダーのテキストを使用しないでください。提供するメッセージの例は、実際に送信する予定のメッセージをできるだけ正確に反映させる必要があります。

6. [Campaign and content attributes] のセクションには、キャンペーンの特定の機能に関連する一連の「はい」または「いいえ」の質問が含まれています。一部の属性は必須であるため、デフォルト値を変更することはできません。

   選択した属性がキャンペーンにとって正確であることを確認してください。

   登録するキャンペーンについて、以下の各項目に該当するかどうかを記入してください。

- 加入者のオプトイン - 受信者は、このキャンペーンに関するメッセージの受信をオプトインできます。
- 加入者のオプトアウト - 受信者は、このキャンペーンに関するメッセージの受信をオプトアウトできます。
- 加入者のヘルプ - ユーザーは HELP キーワードを送信した後、メッセージの送信者に連絡できます。
- 番号プーリング - この 10DLC キャンペーンでは、50以上の電話番号を使用しています。
- ダイレクトレンディングまたはローンアレンジメント - キャンペーンには、直接融資やその他の融資の手配に関する情報が含まれます。
- 埋め込みリンク - 10DLC キャンペーンには、リンクが含まれます。TinyUrl や Bit.ly などの一般的な URL 短縮ツールからのリンクは許可されません。ただし、カスタムドメインを提供する URL 短縮サービスを利用することは可能です。
- Embedded phone number - このキャンペーンには、カスタマーサポート番号ではない電話番号が埋め込まれます。
- アフィリエイトマーケティング - 10DLC キャンペーンには、アフィリエイトマーケティングの情報が含まれます。
- 年齢制限のあるコンテンツ - 10DLC キャンペーンには、通信事業者および Cellular Telecommunications and Internet Association (CTIA) のガイドラインで定義されている年齢制限付きコンテンツが含まれます。

7. [Create] (作成) を選択します。

キャンペーンの登録内容を送信すると、[SMS and voice] ページが表示されます。キャンペーンが送信され、審査中であることを示すメッセージが表示されます。リクエストのステータスは、[10DLC campaigns] タブでご確認いただけます。登録状況は、[10DLC] タブで確認できます。ステータスは、以下のいずれかです。

- アクティブ - 10DLC キャンペーンが承認されました。10DLC 電話番号をリクエストし、その番号を関連付けることができます。詳細については、「Amazon SNS で SMS メッセージング用の 10DLC 番号、通話料無料番号、P2P ロングコードをリクエストする」を参照してください。
- 保留中 - 10DLC キャンペーンはまだ承認されていません。場合によっては、承認に 1 週間以上かかる場合があります。ステータスが変更された場合、Amazon Pinpoint コンソールにその変更が反映されます。ステータスの変更は通知しません。

- 拒否 — 10DLCキャンペーンは拒否されました。詳細については、拒否されたキャンペーンのキャンペーン ID を記入したサポートリクエストを送信してください。
- 停止 - 1 つ以上の通信事業者がお客様の 10DLC キャンペーンを停止しました。詳細については、停止されたキャンペーンのキャンペーン ID を記入したサポートリクエストを送信してください。Amazon Pinpoint では、コンソールに停止理由が表示されません。また、キャンペーンが停止されても通知は送信されません。

8. 10DLC が承認された場合は、そのキャンペーンに関連付ける 10DLC 番号をリクエストできます。10DLC 番号リクエストの詳細については、「Amazon SNS で SMS メッセージング用の 10DLC 番号、通話料無料番号、P2P ロングコードをリクエストする」を参照してください。

複数の AWS リージョンで 10DLC のキャンペーンを利用する

会社を登録すると、その会社はすべての AWS リージョンの AWS アカウントで利用できるようになります。しかし、10DLC のキャンペーンはそうではありません。10DLC キャンペーンは、登録された AWS リージョンでのみ使用できます。

複数の AWS リージョンで 10DLC を利用する場合、それぞれのリージョンで別々の 10DLC キャンペーンを登録する必要があります。このステップは、通信事業者の要件に準拠するために必要です。ユースケースが全く同じでも、キャンペーンを登録するたびに課金されます。

複数のキャンペーンを登録すると、AT&T がキャンペーンごとに 10DLC のスループットレートを提供するため、AT&T を通信事業者として利用している受信者に送信するメッセージのスループットレートが向上するという利点もあります。T-Mobile の 10DLC スループットの扱い方と比較すると、各社 (キャンペーン数に関わらず) 1 日分のメッセージの割り当てが基本となっていることが分かります。

10DLC キャンペーンを編集または削除する

10DLC キャンペーンの HELP レスポンス、STOP レスポンス、サンプルメッセージは、Amazon Pinpoint コンソールを使って編集することができます。さらに、コンソールから 10DLC キャンペーンを削除することも可能です。

10DLC キャンペーンを編集する

キャンペーンが承認されると、HELP、STOP、サンプルメッセージを修正することができます。また、サンプルメッセージを追加することもできます。これらのフィールドの変更は、キャンペーンレジストリや通信事業者からの再承認を必要としません。10DLC キャンペーンが承認された後は、他のフィールドを修正することはできません。

サンプルメッセージは最大 5 つまで登録できます。もともと登録されているサンプルメッセージの数を減らすことはできません。例えば、3 つのサンプル SMS メッセージでキャンペーンを登録した場合、サンプル SMS メッセージの数を 3 つすることはできません。

> **Note**
>
> HELP、STOP、サンプルメッセージ以外のフィールドを変更する場合は、まず 10DLC キャンペーンを削除し、更新した情報を含むキャンペーンを再作成する必要があります。

10DLC キャンペーンを編集するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/pinpoint/ で Amazon Pinpoint コンソールを開きます。
2. ナビゲーションペインで、[SMS and voice] の [Phone numbers] を選択します。
3. [10DLC campaigns] タブで、編集する 10DLC キャンペーンを選択します。
4. [campaign details] ページの [Campaign messages] セクションで、[Edit] を選択します。
5. 以下のフィールドのいずれかを更新します。

   - ヘルプメッセージ
   - ストップメッセージ
   - サンプル SMS メッセージ

   以前に追加されたサンプルメッセージを削除したり、サンプルメッセージの内容を削除してフィールドを空にすることはできません。メッセージの内容を変更せずに削除した場合、更新時に元のメッセージが使用されます。
6. [Update] を選択します。キャンペーンメッセージが更新されたことを知らせる確認バナーが表示されます。

10DLC キャンペーンを削除する

Amazon Pinpoint コンソールを使って、10DLC キャンペーンの一部を削除することができます。10DLC キャンペーンを削除する前に、まずそのキャンペーンに関連するすべての電話番号を削除する必要があります。

> **Important**
>
> キャンペーンから 10DLC 番号を削除すると、その番号へのアクセスはできなくなります。また、一度、削除した 10DLC キャンペーンは元に戻せません。

10DLC キャンペーンを削除するには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/pinpoint/ で Amazon Pinpoint コンソールを開きます。
2. ナビゲーションペインで、[SMS and voice] の [Phone numbers] を選択します。
3. [10DLC campaigns] タブで、編集する 10DLC キャンペーンを選択します。
4. [Phone numbers] セクションで、キャンペーンに関連する電話番号をメモします。
5. [Phone numbers] タブで、削除する 10DLC 番号を選択し、[Remove phone number] を選択します。

   > **Note**
   >
   > このステップは、キャンペーンに複数の 10DLC 電話番号が関連付けられている場合のみ必要です。10DLC キャンペーンに関連する電話番号が 1 つしかない場合、その電話番号が [10DLC campaigns] タブに表示されます。タブに表示される数字をメモします。

6. 確認ボックスで、**`delete`** を入力し、[Confirm] を選択します。[SMS and voice] ページの上部に成功のメッセージが表示されます。
7. キャンペーンに関連する各 10DLC 番号について、先ほどの 2 つのステップを繰り返します。
8. 10DLC キャンペーンに関連する番号を削除した後、[10DLC campaigns] タブを選択します。
9. 削除する 10DLC キャンペーンを選択します。
10. [10DLC campaign details] ページの右上にある [Delete] を選択します。
11. 確認ボックスで、**`delete`** を入力し、[Confirm] を選択します。[SMS and voice] ページの上部に成功のメッセージが表示されます。

ロングコードを 10DLC キャンペーンに関連付ける

既存のロングコードがある場合は、サポートリクエストを提出することで、そのロングコードを現在の 10DLC キャンペーンの 1 つに関連付けることができます。10DLC キャンペーンに関連付ける

ロングコードは、そのキャンペーンでのみ使用でき、他の 10DLC キャンペーンでは使用できません。ロングコードが 10DLC に移行されている間も使用することができます。ただし、承認されるまで、10DLC キャンペーンには使用できません。

リクエストを提出する際には、以下が必要です。

- 10DLC キャンペーンに関連付けるロングコード
- 10DLC キャンペーン ID に関連付けるロングコード

> **Note**
>
> ロングコードをキャンペーンに関連付けるには、その 10DLC キャンペーンを登録しておく必要があります。10DLC キャンペーンをまだ作成して登録していない場合は、「10DLC キャンペーンを登録する」を参照してください。

10DLC にロングコードを割り当てるには

1. AWS Management Console にサインインして、https://console.aws.amazon.com/pinpoint/ で Amazon Pinpoint コンソールを開きます。
2. [設定] で、[SMS と音声] で、[電話番号] タブで次の操作を行います。
3. 10DLC に変換するロングコードを選択します。
4. サポートセンターを開くには、[10DLC キャンペーンに割り当てる] を選択します。
5. ケースタイプで、[サービスの上限の引き上げ] を選択します。
6. [制限のタイプ] で、[Pinpoint] を選択します。
7. [Requests](リクエスト) セクションで [Region](リージョン) を選択し、[Limit](制限) で、[10 DLC - Associate existing US long code to 10DLC campaign](10 DLC - US ロングコードを 10DLC キャンペーンに関連付ける) を選択します。
8. [ケースの説明] の [ユースケースの説明] には、10DLC キャンペーン ID と、そのキャンペーンを関連付けるロングコード番号を必ず記入してください。リクエストには複数のロングコードを記入できますが、キャンペーン ID は 1 つだけ記入してください。
9. [連絡先のオプション] の [優先する連絡先の言語] で、AWS サポートチームとの連絡に使用する言語を選択します。
10. [連絡方法] では、AWS サポートチームとやり取りする方法を指定します。
11. [送信] を選択します。

10DLC クロスアカウントアクセス

10DLC の各電話番号は、単一の AWS リージョンで 1 つのアカウントと関連付けられています。同じ 10DLC の電話番号を使って、複数のアカウントやリージョンでメッセージを送信したい場合、2 つの方法があります。

1. 各 AWS アカウントで同じ会社、同じキャンペーンを登録することができます。これらの登録は、別途管理され、課金されます。複数の AWS アカウントに同じ会社を登録した場合、T-Mobile のお客様に 1 日に送信できるメッセージの数は、それぞれのアカウントで共有されます。
2. 10DLC の登録作業を 1 つの AWS アカウントで完了し、AWS Identity and Access Management (IAM) を使って他のアカウントに 10DLC 番号を通じた送信の許可を与えることができます。

> **Note**
>
> このオプションは、10DLC の電話番号への真のクロスアカウントおよびアクセスを可能にします。ただし、セカンダリアカウントから送信されたメッセージは、プライマリアカウントから送信されたものとして扱われることに注意してください。クォータおよび請求は、セカンダリアカウントではなく、プライマリアカウントに対してカウントされます。

IAM ポリシーによるクロスアカウントアクセスのセットアップ

IAM ロールを使用して、他のアカウントをメインアカウントに関連付けることができます。そして、プライマリーアカウントの 10DLC 番号へのアクセス権をセカンダリーアカウントに付与することで、プライマリーアカウントからのアクセス権をセカンダリーアカウントに委譲することができます。

プライマリーアカウントの 10DLC 番号にアクセスを許可する場合

1. まだの方は、プライマリーアカウントで 10DLC の登録作業を完了させてください。このプロセスには、3つのステップがあります。
   - 会社を登録します。10DLCの使用についての詳細は、会社またはブランドを登録する を参照してください。
   - 10DLC キャンペーン (ユースケース) を登録します。詳細については、「10DLC キャンペーンを登録する」を参照してください。

- 電話番号を 10DLC キャンペーンに関連付けます。詳細については、「ロングコードを 10DLC キャンペーンに関連付ける」を参照してください。

2. 10DLC 電話番号の Publish APIオペレーションを別のアカウントで呼び出せるようにする IAM ロールをプライマリアカウントに作成します。IAM ロールの作成の詳細については、IAM ユーザーガイドのIAM ロールを作成する を参照してください。
3. 10DLC 番号を使用する必要がある他のアカウントで IAM ロールを使用して、プライマリアカウントからアクセス許可を付与し、テストします。例えば、アクセス許可を本番稼働用アカウントから開発用アカウントへアクセス許可を付与します。権利付与とテストの詳細については、IAM ユーザーガイドの IAM ロールを使用した AWS アカウント全体のアクセス権限の委譲を参照してください。
4. 新しいロールを使用して、メインアカウントから 10DLC 番号を使用してメッセージを送信します。ロールの使用についての詳細は、IAM ユーザーガイドの IAM ロールの使用を参照してください。

10DLC の登録に関する情報の取得

会社や 10DLC キャンペーンを登録しようとすると、状況によってはエラーメッセージが表示されることがあります。

会社登録に関する問題

会社登録の際に、Amazon Pinpoint は以下の登録ステータスのいずれかを表示します。確認済みまたは未確認。会社登録のステータスが確認済みであれば、会社登録は成功です。10DLC キャンペーンの作成を開始できます。

会社登録のステータスが未確認の場合、提供された情報に問題があったことになります。Amazon Pinpointコンソールは、会社登録がこのステータスになった理由について、情報を表示します。

10DLC 会社登録の登録事項を表示するには

1. AWS Management Console にサインインして、Amazon Pinpoint コンソール (https://console.aws.amazon.com/pinpoint/) を開きます。
2. ナビゲーションペインで、[SMS] の [電話番号] を選択します。
3. [10DLC campaigns] タブで、キャンペーンの一覧から、詳細情報を表示したい会社名を選択します。

4. [company detail] ページには、登録時に確認された問題が表示されます。[Company info](会社情報) セクションのフィールドに警告マークが表示されている場合は、そのフィールドの情報に登録上の問題があります。

   入力された情報を確認し、すべての項目に正しい情報が含まれていることを確認します。Amazon Pinpoint コンソールから直接、会社登録の情報を編集することができます。会社登録の詳細の変更の詳細については、「登録会社の編集または削除」を参照してください。

キャンペーン登録の問題

10DLC キャンペーンを登録すると、状況によってはエラーメッセージが表示されることがあります。

登録に関する問題を特定できない場合は、AWS Support センター にケースを作成して追加情報を要求することができます。以下の手順に従って、AWS サポートケースを作成します。AWS サポートチームは、10DLC キャンペーン登録が拒否された理由についての情報を提供します。

拒否された 10DLC キャンペーンに関する情報提供のリクエストの送信

1. AWS Management Console (https://console.aws.amazon.com/) にサインインします。
2. [サポート] メニューで [サポートセンター] を選択します。
3. [お客様のサポートケース] ペインで、[ケースを作成] を選択します。
4. [サービス制限の引き上げをご希望ですか?] リンクを選択し、以下を完了します。
   - [制限タイプ] で、[Pinpoint SMS] を選択します。
5. [Requests] で、以下のセクションに入力します。
   - [リージョン] で、キャンペーンを登録しようとした AWS リージョンを選択します。

     > **Note**
     >
     > [Requests] セクションには、リージョンが必要です。また、[Case details] フィールドに入力した場合でも、ここで入力する必要があります。

   - [Resource Type] で、10DLC 登録を選択します。
   - [制限] で、[会社または 10DLC キャンペーン登録拒否] を選択します。
6. [申請する上限数] で、制限タイプの制限の引き上げを選択します。通常、この値は「**1**」です。

7. [ケースの説明]で、拒否された 10DLC キャンペーンの ID を入力します。
8. (オプション) さらにリクエストを送信したい場合は、[Add another request] を選択します。複数のリクエストを含める場合は、それぞれ必要な情報を指定します。必要な情報については、「Amazon SNS で SMS メッセージングのサポートをリクエストする」の他のセクションを参照してください。
9. [連絡先オプション] の [連絡する際の希望言語] で、このケースに関する連絡を受け取る言語を選択します。
10. 完了したら、[送信] を選択します。

通話料無料の番号

通話料無料番号 (TFN) とは、市外局番が 800、888、877、866、855、844、833 のいずれかで始まる 10 桁の番号です。TFN を使用して送信できるのは、トランザクションメッセージのみです。

> **Important**
>
> 米国の携帯電話会社は最近規制を変更し、すべての通話料無料番号 (TFN) を 2022 年 9 月 30 日までに規制機関に登録することを義務付けるようになりました。the section called “通話料無料番号の登録ステータス” にアクセスして TFN のステータスを確認します。会社の登録の詳細については、「the section called “通話料無料番号の登録”」を参照してください。
> 登録は、送信してから処理されるまでに最大 15 営業日かかる場合があります。
> 2023 年 3 月 3 日更新: 2023 年 4 月 1 日より、携帯電話通信事業者は、すべての未登録の通話料無料番号で送信されるメッセージングに対して、業界全体で以下のしきい値を適用します。
>
> - 1 日の制限: 500 件のメッセージ (太平洋時間午前 0 時にリセット)
> - 1 週間の制限: 1,000 件のメッセージ (太平洋時間の日曜午前 0 時にリセット)
> - 1 か月の制限: 2,000 件のメッセージ (太平洋時間の月末午前 0 時にリセット)
>
> 更新 2022 年 9 月 19 日: 2022 年 10 月 1 日より、携帯電話通信事業者は、すべての未登録の通話料無料番号で送信されるメッセージングに対して、次の業界全体のしきい値を適用します:
>
> - 1 日の制限: 2,000 メッセージ
> - 1 週間の制限: 12,000 メッセージ
> - 1 か月の制限: 25,000 メッセージ

できる限り早く登録を完了することをお勧めします。未登録のTFN 経由のメッセージは、ベストエフォート方式で送信されます。通信事業者が未登録のトラフィックを引き続き制限する場合、時間と共にメッセージはさらなるフィルタリングとブロックの対象になります。

トピック

通話料無料の番号の利用ガイドライン

TFN は通常、米国内で登録確認やワンタイムパスワードの送信など、トランザクションメッセージにのみ使用されます。音声メッセージと SMS の両方に使用することができます。平均的なスループットは 3 メッセージパート/秒 (MPS) です。ただし、このスループットは文字エンコーディングに影響されます。文字エンコーディングがメッセージパートに与える影響について詳しくは、Amazon SNS の SMS の文字数制限について を参照してください。TFN の登録の詳細については、「通話料無料番号の登録の要件とプロセス」を参照してください。

顧客アカウントごとに最大 5 つの TFN を持つことができます。1 秒あたり 15 件を超えるが 100 件未満のテキストメッセージを送信する場合は、1 つ以上の 10DLC 送信元 ID を登録することをお勧めします。ユースケースで毎秒 100 件を超えるテキストメッセージを送信する必要がある場合は、1 つ以上のショートコードを購入して登録することをお勧めします。

発信番号として TFN を使用する場合は、次のガイドラインに従ってください。

- サードパーティーの URL 短縮機能で作成された短縮 URL は使用しないでください。これらのメッセージはスパムとしてフィルター処理される可能性が高くなります。

短縮 URL を使用する必要がある場合は、10DLC 番号またはショートコードの使用をご検討ください。ショートコードと 10DLC を使用するには、短縮 URL を指定できるメッセージテンプレートを登録する必要があります。

- キーワードのオプトアウト (STOP) およびオプトイン (UNSTOP) 応答は、通信事業者レベルで設定されることに注意してください。これらのキーワードやその他のキーワードは一切変更できません。また、ユーザーが「STOP」および「UNSTOP」で返信したときに送信されるメッセージを変更することもできません。
- 複数の TFN を使用して、同一または類似のメッセージコンテンツを送信しないでください。通信事業者は、この行為をスノーシューイングまたは番号プールと呼び、これらのメッセージをフィルター処理の対象にします。
- 以下の業界に関連するメッセージはすべて制限対象と見なされ、厳重にフィルタリングまたは完全にブロックされます。これには、制限対象のカテゴリに関連するサービスのワンタイムパスワード (OTP) や多要素認証 (MFA) が含まれます。

  登録が非準拠のユースケースとして拒否され、この指定が間違っていると思われる場合は、サポート経由でリクエストを送信できます。この操作の詳細については、「登録の問題」を参照してください。

  次の表は、制限されるコンテンツの種類を表しています。

カテゴリ	例
ギャンブル	• アプリケーション/ウェブサイト • カジノ • 宝くじ
高リスク金融サービス	• 自動車ローン • Cryptocurrency • 債権回収 • 給料日ローン • 短期高利ローン • 住宅ローン • 学生ローン • 在庫アラート

カテゴリ	例
債権放棄	• 債務統合 • 債務削減 • 信用回復プログラム
G et-rich-quick スキーム	• W ork-from-home プログラム • リスク投資の機会 • ピラミッドまたはマルチレベルのマーケティングスキーム
禁止/規制物質	• 大麻/CBD
フィッシング	• ユーザーに個人情報やウェブサイトのログイン情報を開示させる試み
S.H.A.F.T。	• 性別 • 憎悪 • アルコール • 銃器 • タバコ/ベイプ

通話料無料番号を購入する

TFNsを購入するには、https://console.aws.amazon.com/sms-voice/ の Amazon Pinpoint コンソールを使用します。詳細については、「通話料無料番号の登録の要件とプロセス」を参照してください。

現在、Amazon Pinpoint SMS は、音声メッセージと SMS メッセージの両方で通話料無料番号をサポートしています。Amazon SNS は SMS メッセージングのみをサポートします。

通話料無料番号の登録の要件とプロセス

> **Important**
>
> 指定したユースケース以外の目的に TFN を使用すると、却下される可能性があります。

通話料無料番号の禁止されたユースケース

Amazon SNS では、メッセージがブロックされる場合 (規制薬物やフィッシングに関連するユースケースなど)、または高レベルのフィルタリングが予想される場合 (高リスクの金融メッセージなど)、メッセージを送信する機能が制限されます。通話料無料の番号の利用ガイドライン に定義された制限付きコンテンツのユースケースに関連する TFN は登録できない場合があります。

通話料無料番号の登録

TFN を購入したら、番号を登録する必要があります。これを行う方法については、「Amazon Pinpoint SMS ユーザーガイド」の「通話料無料番号の登録プロセス」を参照してください。

Amazon Pinpoint SMS リージョンでの通話料無料番号のセルフサービス登録

Amazon Pinpoint SMS リージョン で TFN をリクエストした場合は、Amazon Pinpoint SMS ユーザーガイド の「米国の通話料無料番号登録フォーム」に記載されている手順に従って、Amazon Pinpoint SMS コンソールで直接会社登録プロセスを完了します。 Amazon Pinpoint

TFN を登録するときは、情報が完全かつ正確であることを確認してください。そうでないと、登録が却下される場合があります。入力する情報は、会社の本社と完全に一致する必要があります。

Amazon Pinpoint SMS リージョン以外のリージョンでの通話料無料番号の手動フォームベースの登録プロセス

1. この US_TFN_Registration.zip をダウンロードし、サンプル登録フォーム (AWS 米国通話料無料登録フォームビジネス - Final.docx) を使用して、TFN 登録 CSV ファイル (bulkUStfn - Final.csv) に必要な情報を入力します。

   登録リクエストまたはユースケースごとに許容される TFN は最大 5 つです。このルールの免除対象となると思われる場合は、検討のための詳細な説明を提供してください。登録またはユースケースに関連するすべての電話番号を記載してください。
2. AWS サポートでケースを作成します。入力した CSV ファイルをケースに添付し、TFN 登録リクエストを送信します。
3. [ケースを作成] を選択し、[サービス制限の引き上げをご希望ですか?] を選択します。
4. [制限タイプ] で、[SNS テキストメッセージング] を選択します。
5. [Resource Type] (リソースタイプ) で、[10DLC or Toll-free number registration] (10DLC または通話料無料番号登録) を選択します。
6. US_TFN_registration ドキュメントを添付してリクエストを送信してください。

重要な注意事項

1. すべての必要な情報を送信してから、登録が処理されるまでに最大 2 週間かかる場合があります。情報が不足しているか、不完全である場合は、登録手続きが遅れます。登録が拒否された場合は、拒否された理由を確認し、登録できるようにキャンペーンを改善する方法を提案します。
2. TFN は、必要なスループットが限られている多要素認証 (MFA) などのトランザクションユースケースに適しています。各 TFN は 1 秒あたり最大 3 つのテキストメッセージを送信でき、各顧客アカウントは最大 5 つの TFN を持つことができます。テキストメッセージの送信数が 1 秒あたり 15 件を超えるが 100 件未満である場合は、1 つ以上の 10DLC 発信 ID を登録することをお勧めします。ユースケースで毎秒 100 件を超えるテキストメッセージを送信する必要がある場合は、1つ以上のショートコードを購入して登録することをお勧めします。詳細については、「通話料無料の番号の利用ガイドライン」を参照してください。

通話料無料番号の登録ステータス

登録ステータスを確認するには、「Amazon Pinpoint SMS ユーザーガイド」の「登録ステータスを確認する」を参照してください。 Amazon Pinpoint

登録の編集、破棄、削除

Amazon Pinpoint SMS ユーザーガイドを使用して、次のタスクを実行します。

- 登録を編集する
- 登録を破棄する
- 登録を削除する
- 登録リソースを表示する

登録の問題

通話料無料番号の登録が承認されない場合は、拒否の理由を説明するメッセージが表示されます。

通話料無料番号の拒否に関する情報のリクエストを送信するには

1. https://console.aws.amazon.com/ AWS Management Console で にサインインします。
2. [サポート] メニューで [サポートセンター] を選択します。
3. [お客様のサポートケース] ペインで、[ケースを作成] を選択します。
4. [サービス制限の引き上げをご希望ですか?] リンクを選択し、以下を完了します。

- [制限タイプ] で、[Pinpoint SMS] を選択します。

5. [Requests] で、以下のセクションに入力します。

   - [リージョン] で、キャンペーンを登録しようとしたリージョンを選択します。

     > **Note**
     >
     > [Requests] セクションには、リージョンが必要です。また、[Case details] フィールドに入力した場合でも、ここで入力する必要があります。

   - [Resource Type] (リソースタイプ) で、[10DLC or TFN Registration] (10DLC または TFN 登録) を選択します。
   - [制限] で、[会社またはキャンペーンの登録拒否] を選択します。

6. [申請する上限数] で、制限タイプの制限の引き上げを選択します。通常、この値は「**1**」です。
7. (オプション) さらにリクエストを出したい場合は、[Add another request] を選択します。必要な情報については、「Amazon SNS で SMS メッセージングのサポートをリクエストする」の他のセクションを参照してください。
8. [ユースケースの説明] に、拒否された通話料無料番号を入力します。
9. [連絡先オプション] の [連絡する際の希望言語] で、このケースに関する連絡を受け取る言語を選択します。
10. 完了したら、送信 を選択します。

通話料無料番号に関するよくある質問

TFN の登録プロセスに関するよくある質問。

現在、通話料無料番号を所有しているかどうかは、どのようにして確認できますか?

通話料無料番号を所有しているかどうかを確認するには

- https://console.aws.amazon.com/sms-voice/ で Amazon Pinpoint SMS コンソールを開きます。
- ナビゲーションペインで、[SMS] の [電話番号] を選択します。
- TFN のタイプは、通話料無料として表示されます。

通話料無料番号は登録する必要がありますか?

はい。現在所有している TFN を引き続き使用するには、2022 年 9 月 30 日までに登録する必要があります。2022 年 9 月 30 日以降に新しい TFN を購入する場合、メッセージを送信するには登録する必要があります。

通話料無料番号を購入するにはどうすればいいですか?

「Amazon Pinpoint SMS コンソールを使用した電話番号のリクエスト」の指示に従って TFN を購入します。

通話料無料番号を登録するにはどうすればいいですか?

「the section called “通話料無料番号の登録”」の指示に従って TFN を登録してください。

通話料無料番号の登録ステータスはどうなっていますか? そのステータスはどういう意味ですか?

「the section called “通話料無料番号の登録ステータス”」の指示に従って登録とステータスを確認します。

どのような情報を提供すればよいですか?

会社の住所、担当者の連絡先、TFN のユースケースを指定する必要があります。必要な情報は、「the section called “通話料無料番号の登録”」で確認できます。

登録が却下された場合はどうなりますか?

登録が拒否されると、ステータスは [Requires Updates] (更新が必要) に変わります。更新を行うには、「the section called “登録の編集、破棄、削除”」を参照してください。

どのようなアクセス許可が必要ですか?

Amazon Pinpoint SMS コンソールへのアクセスに使用する IAM ユーザー/ロールには、*`#sms-voice:*`*」アクセス許可が必要です。アクセス拒否エラーが表示されます。

通話料無料番号の利点と欠点

利点

通話料無料の発信者は、ロングコードよりも高い MPS を持ち、優れた配信性能を持っています。

欠点

オプトアウトおよびオプトインは、通信事業者レベルで管理されるため、これらを制御することはできません。

メッセージに短縮 URL を含めたり、この番号を使ってプロモーションメッセージを送信したりしないでください。代わりに 10DLC 番号またはショートコードを使用します。ショートコードまたは 10DLC 番号を利用する場合、メッセージテンプレートの登録が必要です。メッセージテンプレートでは、短縮 URL を含めたり、プロモーションメッセージを送信したりできます。ショートコードの詳細については、「ショートコード」を参照してください。10DLC の詳細については、「10DLC」を参照してください。

ショートコード

ショートコードは、通常の電話番号よりも短い数字列です。例えば、米国とカナダでは、標準の電話番号 (ロングコード) は 11 桁、ショートコードは 5 桁または 6 桁の数字で構成されます。Amazon SNS は、専用のショートコードをサポートしています。

専用ショートコード

米国またはカナダの受信者に大量の SMS メッセージを送信する場合は、専用のショートコードを購入することができます。共有プールのショートコードとは異なり、専用ショートコードはお客様専用に予約されています。

利点

覚えやすいショートコードを使用すれば、信頼の構築に役立ちます。ワンタイムパスワードなどの機密情報を送信する必要がある場合は、ショートコードを使用してメッセージを送信することをお勧めします。これにより、顧客は、お客様からのメッセージであることを迅速に判断することができます。

新規顧客獲得キャンペーンを実施している場合は、ショートコードにキーワードを送信するように、見込み顧客を招待することができます (例 : 「サッカーのニュースや情報の取得は「FOOTBALL」というメッセージを「10987」へ送信します」)。ショートコードはロングコードよりも覚えやすいため、顧客がデバイスにショートコードを入力する方が簡単です。顧客がマーケティングプログラムに登録する際に発生する問題を軽減することで、キャンペーンの効果を高めることができます。

携帯端末通信事業者は、アクティブにする前に新しいショートコードを承認する必要があるため、ショートコードから送信されたメッセージに未承諾のフラグが付けられる可能性は低くなります。

ショートコードを使用して SMS メッセージを送信すると、他の種類の送信元アイデンティティを使用する場合よりも、24 時間ごとにさらに多くのメッセージを送信できます。つまり、送信クォータが高くなります。また、1 秒あたりのメッセージのボリュームを大幅に増やすこともできます。つまり、送信レートが高くなります。

欠点

ショートコードを取得するには追加コストがかかるため、実装に時間がかかることがあります。例えば、米国では、ショートコードごとに 1 回限りのセットアップ費用 650 USD と、ショートコードごとに 1 か月当たり 995 USD の追加の繰り返し課金が発生します。すべての通信事業者ネットワークで、ショートコードを有効にするには、8〜12 週間かかります。別の国やリージョンの料金とプロビジョニング時間を確認するには、「Amazon SNS による SMS メッセージングの専用ショートコードをリクエストする」で説明されている手順を実行します。

パーソントゥーパーソン (P2P) ロングコード

> **Important**
>
> 2023 年 8 月 31 日より、米国および米国領 (プエルトリコ、グアム、米領サモア諸島、および US バージン諸島)に SMS テキストメッセージを送信する際は、10DLC 番号または通話無料番号などの専用番号が必要となります。これらの地域の口ケーションとして米国を使用している場合、ロングコードのリクエストは拒否されます。

> **Important**
>
> 2021 年 6 月 1 日より、米国のテレコムプロバイダーは、米国宛の Application-to-Person (A2P) 通信に Person-to-Person (P2P) ロングコードの使用をサポートしていません。したがって、これらのメッセージには別のタイプの送信元 ID を使用する必要があります。詳細については、「10DLC」を参照してください。

P2P ロングコードは、受取人が居住する国またはリージョンの番号形式を使用する電話番号です。また、P2P ロングコードは、ロング番号または仮想携帯電話番号とも呼ばれます。例えば、米国やカナダでは、P2P ロングコードは、1 桁の国コード、3 桁の市外局番、7 桁の電話番号の 11 桁 で構成されます。

P2P ロングコードのリクエストについては、「Amazon SNS で SMS メッセージング用の 10DLC 番号、通話料無料番号、P2P ロングコードをリクエストする」を参照してください。

利点

専用の P2P ロングコードは、Amazon SNS アカウントのみが使用するように予約されています。そのため、他のユーザーと共有されることはありません。専用の P2P ロングコードを使用する場

合は、各メッセージを送信する際に使用する P2P ロングコードを指定することができます。複数のメッセージを同じ顧客に送信する場合は、各メッセージが同じ電話番号から送信されるように表示することができます。そのため、専用の P2P ロングコードは、ブランドまたはアイデンティティを確立する上で便利です。

欠点

P2P ロングコードは、US の送信先への A2P 通信ではサポートされていません。

専用の P2P ロングコードから、1 日あたり数百通のメッセージを送信すると、モバイルキャリアにより、お使いの番号が未承諾メッセージを送信する番号として認識される場合があります。P2P ロングコードにフラグが設定されると、メッセージが受取人に配信されない場合があります。

P2P ロングコードのスループットも制限されています。最大送信レートは国によって異なります。詳細については、AWS サポートにお問い合わせください。大量の SMS メッセージを送信する予定がある場合、または 1 秒あたり 2 通以上のメッセージを超えるレートで送信する場合は、専用のショートコードを購入する必要があります。

米国を含め、一部の通信事業者は、A2P SMSメッセージを送信するために P2P ロングコードを使用することを許可していません。A2P SMS は、アプリケーションに携帯電話番号を入力してモバイルデバイスに送信するメッセージです。A2P メッセージは、マーケティングメッセージ、ワンタイムパスワード、予定のリマインダーなどの一方向の会話です。A2P メッセージを送信する予定の場合は、専用ショートコード (米国またはカナダにいる場合) を購入するか、送信者 ID (送信者 ID がサポートされている国またはリージョンに受信者がいる場合) を使用する必要があります。

10DLC 番号は、米国内のメッセージ送信にのみ使用されます。10DLC 番号を利用するには、企業ブランドと番号を関連付けるキャンペーンを登録する必要があります。承認されると、[SMS and voice] ページの Amazon Pinpoint コンソール (https://console.aws.amazon.com/pinpoint/) から 10DLC の電話番号をリクエストすることができます。申請後、承認されるまでの期間は 7～10 日間です。この番号は他のキャンペーンとの併用はできません。

米国製品番号の比較

この表は、米国の電話番号タイプのサポート比較を示しています。

製品の特徴	ショートコード	通話料無料の番号	10DLC	
数値の書式	5～6 桁	10 桁の番号	10 桁の番号	

製品の特徴	ショートコード	通話料無料の番号	10DLC	
サポートチャネル	SMS	SMS	SMS	
SMS トラフィックの種類	プロモーションとトランザクション	トランザクション	プロモーションとトランザクション	
要検査	はい	いいえ	はい	
予想プロビジョニング時間	12 週間[1]	15 営業日	1 週間	
SMS スループット (1 秒あたりの SMS メッセージ数) [2]	1 秒あたり 100 件のメッセージパート。より高いスループットを有料で利用できます。	1 秒あたり 3 件のメッセージ	10DLC 登録によって異なります。1 秒間に最大 100 個のメッセージパートをサポートします。	
必須キーワード	オプトイン、オプトアウト、HELP	STOP、UNSTOP。これらはネットワークで管理されます。オプトアウトおよびオプトインメッセージを変更することはできません。	オプトイン、オプトアウト、HELP	

[1] プロビジョニングの見積もりには承認時間が含まれていません。

[2] SMS メッセージの最大サイズの詳細については、「携帯電話に発行する」を参照してください。

Amazon SNS で SMS メッセージングのサポートをリクエストする

Amazon SNS での特定の SMS オプションは、AWS Support に問い合わせるまで AWS アカウントでご利用いただけます。[AWS Support センター] でケースを開き、以下のいずれかをリクエストします。

- 毎月の SMS 利用額しきい値の増加

  デフォルトでは、毎月の利用額しきい値は 1.00 USD です。利用額しきい値により、Amazon SNS で送信できるメッセージのボリュームが決まります。SMS ユースケースで予想される毎月のメッセージ量に合う利用額しきい値をリクエストします。
- [SMS サンドボックス] からの削除すると、制限なくSMSメッセージを送信できます。詳細については、「SMS サンドボックス外への移動」を参照してください。
- 専有の発信元番号
- 専用送信者 ID

  送信者 ID は、受信者のデバイスで送信者として表示されるカスタム ID です。例えば、メッセージソースを識別しやすいように、ビジネスブランドを使用できます。送信者 ID のサポートについては、国または地域によって異なります。詳細については、「サポートされている国と地域」を参照してください。

AWS Support センターでケースを作成するときは、送信するリクエストのタイプに必要なすべての情報を含めてください。含めない場合、続行する前に AWS Support がお客様に連絡してこの情報を取得します。詳細なケースを送信することにより、ケースが遅れなく処理されます。特定のタイプの SMS リクエストに必要な詳細については、以下のトピックを参照してください。

トピック

- Amazon SNS による SMS メッセージングの専用ショートコードをリクエストする
- Amazon SNS で SMS メッセージング用の 10DLC 番号、通話料無料番号、P2P ロングコードをリクエストする
- Amazon SNS で SMS メッセージングの送信者 ID をリクエストする
- Amazon SNS の毎月の SMS 使用クォータの引き上げをリクエストする

Amazon SNS による SMS メッセージングの専用ショートコードをリクエストする

ショートコードは、大量の SMS メッセージの送信に使用できる数字です。ショートコードは、application-to-person (A2P) メッセージング、2 要素認証 (2FA)、マーケティング用に頻繁に使用されます。通常、ショートコードには、拠点とする国または地域に応じて 3~7 桁の数字が含まれます。

ショートコードは、ショートコードが拠点とする同じ国での受信者へのメッセージの送信にのみ使用できます。ユースケースで複数の国のショートコードを使用する必要がある場合、受信者が居住する国ごとに個別のショートコードをリクエストする必要があります。

ショートコードの料金については、「Amazon SNS 料金表」を参照してください。

> **Important**
>
> Amazon SNS で SMS メッセージングを初めて使用する場合、予想される SMS ユースケース需要を満たす毎月の SMS 利用額しきい値をリクエストしてください。デフォルトでは、毎月の利用額しきい値は 1.00 USD です。ショートコードのリクエストが含まれているのと同じサポートケースで、利用額しきい値の増額をリクエストできます。または、別のケースを使用することもできます。詳細については、「Amazon SNS の毎月の SMS 使用クォータの引き上げをリクエストする」を参照してください。
> さらに、保護された健康情報 (PHI) が含まれるメッセージを送信するための専用のショートコードをリクエストする場合は、以下に説明するように、サポートケースを開くときに [ケースの説明] でこの目的を明示する必要があります。

Amazon SNS ショートコードのサポートケースを開く

以下のステップを実行して、AWS Support でケースを開きます。

> **Note**
>
> リクエストフォームの一部のフィールドは「オプション」としてマークされています。ただし、AWS Support では、リクエストを処理するために次のステップで説明されているすべての情報が必要です。必要な情報をすべて入力しないと、リクエストの処理に遅延が発生する場合があります。

専用のショートコードをリクエストするには

1. AWS サポートセンターに移動します。
2. AWS Management Console (https://console.aws.amazon.com/) にサインインします。
3. [サポート] メニューで [サポートセンター] を選択します。
4. [お客様のサポートケース] ペインで、[ケースを作成] を選択します。
5. [サービス制限の引き上げをご希望ですか?] リンクを選択し、以下を完了します。
   - [制限タイプ] で、[SNS テキストメッセージング] を選択し、以下を完了します。
   - (オプション) [SMS メッセージを送信するサイトまたはアプリへのリンクを指定] で、利用者が SMS メッセージの受信をオプトインするサイトまたはアプリケーションの名前へのリンクを指定します。
   - (オプション) [What type of messages do you plan to send (送信するメッセージのタイプ)] で、ロングコードを使用して送信する予定のメッセージのタイプを選択します。
     - [ワンタイムパスワード] - ウェブサイトまたはアプリケーションを認証するために顧客が使用するパスワードを提供するメッセージ。
     - [プロモーション]- 特価販売やお知らせなど、ビジネスやサービスを宣伝する非クリティカルなメッセージ。
     - [トランザクション]- 注文確認やアカウントアラートなど、顧客のトランザクションをサポートする重要な情報メッセージ。トランザクションメッセージにプロモーションコンテンツまたはマーケティングコンテンツを含めることはできません。
   - (オプション) [Which AWS Region will you be sending messages from] で、メッセージを送信するリージョンを選択します。
   - (オプション) [メッセージの送信先となる国] で、SMS メッセージを送信する先の国または地域を入力します。
   - (オプション) [お客様がメッセージの受信をオプトインする方法] で、お客様がメッセージの受信をオプトインする方法の説明を入力します。
   - (オプション) [お客様にメッセージを送信するために使用するメッセージテンプレートを指定してください] で、使用するテンプレートを指定します。
6. [Requests] で、以下のセクションに入力します。
   - [リージョン] で、ショートコードリクエストの AWS リージョンを選択します。

> **Note**
>
> [Requests] セクションには、リージョンが必要です。また、[Case details] フィールドに入力した場合でも、ここでも入力する必要があります。

- [リソースタイプ] で、[専用 SMS ショートコード] を選択します。
- [制限] で、お客様のユースケースに最も近いオプションを選択します。

7. [申請する上限数] で、リクエストする送信者 ID の数を選択します。通常、この値は「**1**」です。
8. (オプション) さらにリクエストを出したい場合は、[Add another request] を選択します。必要な情報については、「Amazon SNS で SMS メッセージングのサポートをリクエストする」の他のセクションを参照してください。
9. [ケースの説明] で、ユースケースの概要を入力し、受信者がショートコードで送信されるメッセージにサインアップする方法を含め、次の情報を指定します。

   - 会社情報:
     - 会社名
     - 会社の郵送先住所
     - リクエストの主な連絡先の名前と電話番号
     - 会社におけるサポートの E メールアドレスと通話料無料の番号
     - 会社の税 ID
     - 製品またはサービスの名称

   - ユーザーのサインアッププロセス:
     - 会社のウェブサイト、または顧客がショートコードからメッセージを受信するためにサインアップするウェブサイト。
     - ユーザーがショートコードからメッセージを受信するためにサインアップする方法。以下のうち 1 つ以上を指定します。
       - **`Text messages`**
       - **`Website`**
       - **`Mobile app`**
       - **`Other`** その他の場合は、説明してください。

- ウェブサイト、アプリ、または他の場所でメッセージにサインアップするためのオプションのテキスト。
- 二重オプトインに使用する予定の一連のメッセージ。次のすべての情報を入力します。
  1. ユーザーがサインアップしたときに送信する予定の SMS メッセージ このメッセージは、定期的なメッセージに対するユーザーの同意を求めます。次に例を示します。ExampleCorp: アカウントトランザクションアラートを受け取るには、YES と返信してください。メッセージ & データレートが適用される場合があります。
  2. ユーザーに求めるオプトインレスポンス。これは、通常 YES などのキーワードです。
  3. 顧客がこのキーワードをショートコードに送信したときに送信する確認メッセージ。次に例を示します。ExampleCorp からのアカウントアラートに登録されました。メッセージ & データレートが適用される場合があります。キャンセルするには STOP、詳細を参照するには HELP と送信してください。

- メッセージの目的:
  - ショートコードを使用して送信する予定のメッセージの目的。以下のいずれかのオプションを指定します。
    - **Promotions and marketing**
    - **Location-based services**
    - **Notifications**
    - **Information on demand**
    - **Group chat**
    - **Two-factor authentication (2FA)**
    - **Polling and surveys**
    - **Sweepstakes or contests**
    - **Other** その他の場合は、説明してください。
  - 自身のビジネス以外のビジネスのプロモーションまたはマーケティングメッセージの送信にショートコードを使用する予定があるかどうか。
  - 医療保険の携行性と責任に関する法律 (HIPAA) および関連する法規で定義されているように、保護された健康情報 (PHI) を含むメッセージの送信にショートコードを使用するかどうか。

- メッセージの内容:

- 顧客が特定のキーワードを送信してメッセージにオプトインする際に送信する予定のメッセージです。このキーワードやメッセージを指定する場合はご注意ください。このメッセージを変更するのに数週間かかることがあります。ショートコードを作成する場合、キーワードやメッセージは、ショートコードを使用している国の携帯電話キャリアで登録されます。メッセージは、次の例のように表示されます。*ProductName* アラートにようこそ。メッセージ & データレートが適用されます。*2* メッセージ/月。ヘルプが必要な場合は HELP、キャンセルの場合は STOP と返信してください。
- 顧客がキーワード HELP でメッセージに応答したときに送信する応答。このメッセージには、カスタマーサポートのお問い合わせ先情報を含める必要があります。次に例を示します。*ProductName* アラート: ヘルプについては *example.com/help* を参照するか、*(800) 555-0199* までお問い合わせください。メッセージ & データレートが適用されます。*2* メッセージ/月。キャンセルするには STOP と返信してください。
- 顧客がキーワード STOP でメッセージに応答したときに送信する応答。このメッセージは、ユーザーが今後メッセージを受信しないことを確認する必要があります。次に例を示します。*ProductName* アラートよりサブスクリプション解除されます。今後メッセージは送信されません。ヘルプが必要な場合は HELP と返信するか、*(800) 555-0199* までお問い合わせください。
- ユーザーがメッセージにサブスクライブしていることの定期的なアラームとして送信する予定のテキスト。次に例を示します。リマインダー: ExampleCorp からのアカウントアラートにサブスクライブしています。メッセージ & データレートが適用される場合があります。キャンセルするには STOP、詳細を参照するには HELP と送信してください。
- ショートコードを使用して送信する予定のメッセージの各タイプの例。少なくとも 3 つ以上の例を入力します。3 つ以上のタイプのメッセージを送信する場合は、すべての例を入力します。

> **Important**
>
> 携帯端末通信事業者では、ショートコードを設定するために上記のすべての情報を入力する必要があります。この情報をすべて提供するまで、リクエストを処理できません。

10. (オプション) さらにリクエストを送信したい場合は、[Add another request] を選択します。複数のリクエストを含める場合は、それぞれ必要な情報を指定します。必要な情報については、「Amazon SNS で SMS メッセージングのサポートをリクエストする」の他のセクションを参照してください。

11. [連絡先オプション] の [連絡する際の希望言語] で、このケースに関する連絡を受け取る言語を選択します。
12. 完了したら、[送信] を選択します。

リクエストの受信後、24 時間以内に最初の応答を提供します。追加の情報をご依頼することがあります。ショートコードを提供できる場合は、リクエストで指定した国または地域のショートコードの取得に関連付けられている利用料金に関する情報が送信されます。また、国または地域のショートコードのプロビジョニングに必要な時間の見積もりも提供されます。ショートコードのプロビジョニングは、通常数週間かかりますが、ショートコードが拠点とする国または地域に応じてこの所要時間がかなり短縮されたり、さらに時間がかかったりする可能性があります。

> **Note**
>
> ショートコードの使用に関連する料金は、通信業者にショートコードリクエストを開始した直後から開始されます。ショートコードのプロビジョニングがまだ完了していない場合でも、これらの料金を支払う責任はお客様にあります。

迷惑なコンテンツや悪意のあるコンテンツを送信するためにシステムが悪用されないように、各リクエストを慎重に検討する必要があります。お客様のユースケースが当社の方針と一致しない場合は、リクエストを承認できない場合があります。

次のステップ

携帯電話キャリアにショートコードを登録し、Amazon SNS コンソールで設定を確認しました。これで、Amazon SNS を使用し、発信元番号としてショートコードを使用して SMS メッセージを送信できるようになりました。

Amazon SNS で SMS メッセージング用の 10DLC 番号、通話料無料番号、P2P ロングコードをリクエストする

> **Important**
>
> 2021 年 6 月 1 日以降、米国の通信事業者は、米国の宛先への person-to-person (A2P) 通信の application-to-person (P2P) ロングコードの使用をサポートしなくなります。したがって、これらのメッセージには別のタイプの送信元 ID を使用する必要があります。詳細については、「10DLC」を参照してください。

10DLC 番号をリクエストするには、通話料無料の番号と P2P ロングコード、Amazon Pinpoint コンソールを使用します。詳細な手順については、『Amazon Pinpoint ユーザーガイド』の「番号をリクエストする」を参照してください。

> **Important**
>
> 米国の携帯電話会社は最近規制を変更し、すべての通話料無料番号 (TFN) を 2022 年 9 月 30 日までに規制機関に登録することを義務付けるようになりました。通話料無料番号の登録の詳細については、「通話料無料番号の登録」を参照してください。
> 2022 年 9 月 30 日以前に通話料無料番号を購入した場合、そのステータスは 2022 年 10 月 1 日までアクティブ状態になります。ただし、その間に登録を完了すると、登録は完了済み状態に設定されて戻されます。それ以外の場合は保留中状態となり、番号を登録するか、登録が戻されるか、登録がアクティブ状態に設定されるまで、メッセージを送信することはできません。
> 登録には最大 15 営業日かかる場合があります。

10DLC の会社とキャンペーンを登録するには、米国東部 (バージニア北部) リージョンで Amazon Pinpoint コンソールを開きます。10DLC 番号をリクエストする代わりに、AWS Service Quotas コンソールを使用して、そのリージョンの 10DLC 番号をリクエストしながらサービス制限の引き上げケースを作成します。Amazon Pinpoint を利用できるリージョンについての詳細は、「AWS 全般のリファレンス」の「Amazon Pinpoint エンドポイントとクォータ」を参照してください。

> **Note**
>
> Amazon SNS で SMS メッセージングを初めて使用する場合、予想される SMS ユースケース需要を満たす毎月の SMS 利用額しきい値もリクエストする必要があります。デフォルトでは、毎月の利用額しきい値は 1.00 USD です。詳細については、「Amazon SNS の毎月の SMS 使用クォータの引き上げをリクエストする」を参照してください。

Amazon SNS で SMS メッセージングの送信者 ID をリクエストする

> **Important**
>
> Amazon SNS で SMS メッセージングを初めて使用する場合、予想される SMS ユースケース需要を満たす毎月の SMS 利用額しきい値をリクエストしてください。デフォルトでは、毎月の利用額しきい値は 1.00 USD です。送信者 ID のリクエストが含まれているのと同じサ

ポートケースで、利用額しきい値の増額をリクエストできます。または、必要に応じて、別のケースを使用することもできます。詳細については、「Amazon SNS の毎月の SMS 使用クォータの引き上げをリクエストする」を参照してください。

SMS メッセージングでは、送信者 ID は、受信者のデバイスにメッセージ送信者として表示される名前です。送信者 ID は、メッセージの受信者に対して自分を識別するのに役立ちます。

送信者 ID のサポートについては、国または地域によって異なります。例えば、米国の通信事業者は送信者 ID をまったくサポートしていませんが、インドの通信事業者では送信者が送信者 ID を使用することが求められています。送信者 ID をサポートしている国の詳細な一覧は、「サポートされている国と地域」を参照してください。

Important

一部の国では、メッセージの送信に使用する前に、送信者 ID を登録する必要があります。国または地域によっては、この登録処理には数週間かかる場合があります。事前登録された送信者 ID を必要とする国は、「サポートされている国」ページの表に示されています。SMS メッセージの複数の AWS アカウントで同じ送信者 ID を使用して登録できます。エンタープライズサポートがあり、複数のテンプレートを複数のアカウントにまたがって登録しておられる場合は、以下のステップに従って、テクニカルアカウントマネージャーと協力して、オンボーディングエクスペリエンスを調整してください。

送信者 ID がサポートされている国または地域の受信者にメッセージを送信し、その国または地域で送信者 ID を登録する必要がない場合、追加のステップを実行する必要はありません。送信者 ID 値を含むメッセージの送信をすぐに開始できます。

送信者 ID の登録が必要な国または地域にメッセージを送信する場合は、このページの手順を完了する必要があります。

ステップ 1: Amazon SNS SMS ケースを開く

送信者 ID が必要な国の受信者にメッセージを送信する予定の場合は、AWS サポートセンターで新しいケースを作成して送信者 ID をリクエストできます。

> **Note**
>
> 送信者 ID が許可されているが必須ではない国の受信者にメッセージを送信する場合は、サポートセンターでケースを開く必要はありません。送信者 ID を使用するメッセージの送信をすぐに開始できます。

送信者 ID をリクエストするには

1. AWS Management Console (https://console.aws.amazon.com/) にサインインします。
2. [サポート] メニューで [サポートセンター] を選択します。
3. [お客様のサポートケース] ペインで、[ケースを作成] を選択します。
4. [サービス制限の引き上げをご希望ですか?] リンクを選択し、以下を完了します。

   - [制限タイプ] で、[Pinpoint SMS] を選択します。
   - (オプション) [SMS メッセージを送信する先のサイトまたはアプリへのリンクを指定] で、利用者が SMS メッセージの受信をオプトインするウェブサイトまたはアプリケーションを指定します。
   - (オプション) [What type of messages do you plan to send (送信するメッセージのタイプ)] で、ロングコードを使用して送信する予定のメッセージのタイプを選択します。
     - [ワンタイムパスワード] - ウェブサイトまたはアプリケーションを認証するために顧客が使用するパスワードを提供するメッセージ。
     - [プロモーション]- 特価販売やお知らせなど、ビジネスやサービスを宣伝する非クリティカルなメッセージ。
     - [トランザクション]- 注文確認やアカウントアラートなど、顧客のトランザクションをサポートする重要な情報メッセージ。トランザクションメッセージにプロモーションコンテンツまたはマーケティングコンテンツを含めることはできません。
   - (オプション) [メッセージの送信元の AWS リージョン] で、SMS メッセージの送信元の AWS リージョン リージョンを選択します。
   - (オプション) [メッセージの送信先となる国] で、送信者 ID を登録する国を入力します。送信者 ID のサポートと送信者 ID 登録要件は、国によって異なります。詳細については、「サポートされている国と地域」を参照してください。

     国のリストが、このテキストボックスの制限文字数を超えている場合は、代わりに [ケースの説明] セクションで国を一覧表示できます。

- (オプション) [お客様がメッセージの受信をオプトインする方法] で、お客様がメッセージの受信をオプトインする方法の説明を入力します。
- (オプション) [お客様にメッセージを送信するために使用するメッセージテンプレートを指定してください] で、使用するメッセージテンプレートを指定します。

5. [Requests] で、以下のセクションに入力します。

   - [リージョン] で、送信者 ID の [AWS リージョン] を選択します。

     > **Note**
     >
     > [Requests] セクションには、リージョンが必要です。また、[Case details] フィールドに入力した場合でも、ここで入力する必要があります。

   - [リソースタイプ] で、[一般的な制限] を選択します。
   - [制限] で、[SMS の本番稼働用アクセス] を選択します。

6. [申請する上限数] で、リクエストする送信者 ID の数を選択します。通常、この値は「**1**」です。
7. (オプション) さらにリクエストを出したい場合は、[Add another request] を選択します。必要な情報については、「Amazon SNS で SMS メッセージングのサポートをリクエストする」の他のセクションを参照してください。
8. [Case description] の [Use case description] で、次の詳細を入力します。

   - 登録する送信者 ID。
   - SMS メッセージに使用する予定のテンプレート。
   - 各受信者に 1 か月あたりに送信する予定のメッセージの数。
   - 顧客がお客様からのメッセージの受信をオプトインする方法に関する情報。
   - 会社または組織の名前。
   - 会社または組織に関連付けられている住所。
   - 会社または組織の拠点を置く国。
   - 会社または組織の電話番号。
   - 会社または組織のウェブサイトの URL。

9. [連絡先オプション] の [連絡する際の希望言語] で、このケースに関する連絡を受け取る言語を選択します。
10. 完了したら、[送信] を選択します。

リクエストの受信後、24 時間以内に最初の応答を提供します。追加の情報をご依頼することがあります。送信者 ID を提供できる場合は、送信者 ID のプロビジョニングに必要な時間の見積もりを送信します。

迷惑なコンテンツや悪意のあるコンテンツを送信するためにシステムが悪用されないように、各リクエストを慎重に検討する必要があります。お客様のユースケースが当社の方針と一致しない場合は、リクエストを承認できない場合があります。

ステップ 2: Amazon SNS コンソールで SMS 設定を更新する

送信者 ID の取得プロセスを完了すると、ケースに応答します。この通知を受信した場合、このセクションの手順を実行して、アカウントを使用して送信されるすべてのメッセージの送信者 ID をデフォルトの送信者 ID として使用するように Amazon SNS を設定します。または、メッセージの発行時に使用する送信者 ID を指定することもできます。

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[モバイル]、[テキストメッセージング (SMS)] の順に選択します。
3. [テキストメッセージングの優先設定] セクションで、[編集] を選択します。
4. [詳細] セクションの [デフォルトの送信者 ID] フィールドに、アカウントからのすべてのメッセージのデフォルトとして使用する送信者 ID を入力します。
5. 変更が完了したら、[変更の保存] を選択します。

次のステップ

送信者 ID を登録し、Amazon SNS コンソールで設定を更新しました。これで Amazon SNS を使用して、送信者 ID で SMS メッセージを送信できるようになりました。サポートされている国の SMS 受信者のデバイスにメッセージ送信者として送信者 ID が表示されます。メッセージの公開時に別の送信者 ID が使用されている場合、ここで設定されているデフォルトの ID は上書きされます。

Amazon SNS の毎月の SMS 使用クォータの引き上げをリクエストする

Amazon SNS では、アカウントを使用して SMS を送信することで発生する月あたりの最大コストを管理するのに役立つ使用クォータを提供しています。使用クォータは、悪意のある攻撃を受けた場合にリスクを抑え、アップストリームアプリケーションが想定外に多くのメッセージを送信することを防ぎます。Amazon SNS が SMS メッセージの送信により、当月の使用クォータを超えるコストが発生すると判断した場合、SMS メッセージの発行を停止するよう設定できます。

運用に影響が及ばないように、使用クォータが本番用ワークロードをカバーできるよう十分高いクォータをリクエストすることをお勧めします。詳細については、「ステップ 1: Amazon SNS SMS

ケースを開く」を参照してください。クォータを受け取った後、「ステップ 2: SMS 設定を更新する」の説明に従って、フルクォータまたはより小さい値を適用して、リスクを管理できます。より小さい値を適用することで、毎月のコストを制御できます。そのためには、必要に応じてスケールアップするオプションを使用します。

> **Important**
>
> Amazon SNS は分散システムであるため、使用クォータを超えると数分以内に SMS メッセージの送信を停止します。この間に SMS メッセージを送信し続けると、クォータを超えるコストが発生する可能性があります。

すべての新しいアカウントに対して 1 か月あたり 1.00 USD で使用クォータを設定します。このクォータは、Amazon SNS のメッセージ送信機能をテストできるようにすることを目的としています。アカウントの SMS 使用クォータの引き上げをリクエストするには、AWS サポートセンターでクォータ増加ケースを開きます。

ステップ 1: Amazon SNS SMS ケースを開く

毎月の使用クォータの増加をリクエストするには、AWS サポートセンターでクォータ増加ケースを開きます。

> **Note**
>
> リクエストフォームの一部のフィールドは「オプション」としてマークされています。ただし、AWS Support では、リクエストを処理するために次のステップで説明されているすべての情報が必要です。必要な情報をすべて入力しないと、リクエストの処理に遅延が発生する場合があります。

1. AWS Management Console (https://console.aws.amazon.com/) にサインインします。
2. [サポート] メニューで [サポートセンター] を選択します。
3. [お客様のサポートケース] ペインで、[ケースを作成] を選択します。
4. [サービス制限の引き上げをご希望ですか?] リンクを選択して、以下を完了します。
   - [制限タイプ] で、[SNS テキストメッセージング] を選択します。

- (オプション) [Provide a link to the site or app which will be sending SMS messages (SMS メッセージを送信するサイトまたはアプリケーションへのリンクを指定する)] で、SMS メッセージを送信する Web サイト、アプリケーション、またはサービスに関する情報を入力します。
- (オプション) [What type of messages do you plan to send (送信するメッセージのタイプ)] で、ロングコードを使用して送信する予定のメッセージのタイプを選択します。
  - [ワンタイムパスワード] - ウェブサイトまたはアプリケーションを認証するために顧客が使用するパスワードを提供するメッセージ。
  - [プロモーション]- 特価販売やお知らせなど、ビジネスやサービスを宣伝する非クリティカルなメッセージ。
  - [トランザクション]- 注文確認やアカウントアラートなど、顧客のトランザクションをサポートする重要な情報メッセージ。トランザクションメッセージにプロモーションコンテンツまたはマーケティングコンテンツを含めることはできません。
- (オプション) [Which AWS Region will you be sending messages from] で、メッセージを送信するリージョンを選択します。
- (オプション) [Which countries do you plan to send messages to] で、ショートコードを購入する国または地域を入力します。
- (オプション) [How do your customers opt to receive messages from you] で、オプトインプロセスの詳細を入力します。
- (オプション) [How do your customers opt to receive messages from you] で、オプトインプロセスの詳細を入力します。[Please provide the message template that you plan to use to send messages to your customers] の項目に使用するテンプレートを入力します。

5. [Requests] で、以下のセクションに入力します。

   - [Region] で、メッセージの送信元となるリージョンを選択します。

     > **Note**
     >
     > [Requests] セクションには、リージョンが必要です。また、[Case details] フィールドに入力した場合でも、ここで入力する必要があります。

   - [リソースタイプ] で、[一般的な制限] を選択します。
   - [制限] で、[Account Spend Threshold Increase] を選択します。
6. [New limit value] に、毎月の SMS メッセージに使用する上限額を USD で入力します。
7. [Case description] の [Use case description] で、次の詳細を入力します。

- SMS メッセージを送信する会社またはサービスのウェブサイトやアプリ。
- ウェブサイトやアプリによって提供されるサービスと、SMS メッセージがそのサービスにどのように貢献するか。
- ユーザーがウェブサイト、アプリ、他の場所で任意の SMS メッセージ受信にサインアップする方法。

リクエストした使用クォータ ([New quota value] で指定した値) が 10,000 USD を超えた場合、メッセージを送信する国ごとに以下の追加の詳細を指定します。

- 送信者 ID とショートコードのどちらを使用しているか。送信者 ID を使用している場合は、次の情報を指定します。
  - 送信者 ID。
  - 送信者 ID がその国の携帯電話キャリアに登録されているかどうか。
- メッセージングの予想される 1 秒あたりの最大トランザクション数 (TPS)。
- 平均メッセージサイズ。
- 国に送信するメッセージのテンプレート。
- (オプション) 文字エンコードのニーズ (ある場合)。

8. (オプション) さらにリクエストを送信したい場合は、[Add another request] を選択します。複数のリクエストを含める場合は、それぞれ必要な情報を指定します。必要な情報については、「Amazon SNS で SMS メッセージングのサポートをリクエストする」の他のセクションを参照してください。
9. [連絡先オプション] の [連絡する際の希望言語] で、このケースに関する連絡を受け取る言語を選択します。
10. 完了したら、[送信] を選択します。

AWS Support チームは、お客様のリクエストに対して、24 時間以内に最初の回答を行います。

迷惑なコンテンツや悪意のあるコンテンツを送信するためにシステムが悪用されないように、私たちは各リクエストを慎重に検討しています。可能であれば、24 時間以内にリクエストを承認します。ただし、追加情報が必要な場合は、お客様のリクエストの解決に時間がかかる場合があります。

お客様のユースケースが私たちのポリシーに一致しない場合は、リクエストを承認できない場合があります。

ステップ 2: Amazon SNS コンソールで SMS 設定を更新する

毎月の使用クォータの引き上げが通知された後、Amazon SNS コンソールのアカウントで使用クォータを調整する必要があります。

> **Important**
>
> 次のステップを完了する必要があります。完了しない場合、SMS の利用限度額は増えません。

コンソールで使用クォータを調整するには

1. Amazon SNS コンソールにサインインします。
2. 左側のナビゲーションメニューを開き、[モバイル] を展開し、[テキストメッセージング (SMS)] を選択します。
3. [モバイルテキストメッセージング (SMS)] ページの [テキストメッセージプリファレンス] セクションで、[編集] を選択します。
4. [テキストメッセージングの優先設定の編集] ページの [詳細] で、[アカウントの使用制限] に新しい SMS 使用限度額を入力します。

   > **Note**
   >
   > デフォルトの使用制限より高い値を入力すると、警告が表示される場合があります。このテキストは無視して構いません。

5. [Save changes] (変更の保存) をクリックします。

   > **Note**
   >
   > 「Invalid Parameter」というエラーが表示された場合は、AWS サポートの連絡先を確認し、新しい SMS 使用制限が正しく入力されていることを確認します。それでも問題が解決しない場合は、AWS サポートセンターでケースを開きます。

SMS メッセージプリファレンスを設定する

Amazon SNS を使用して、SMS メッセージングのプリファレンスを指定します。例えば、配信をコストまたは信頼性に対して最適化するかや、毎月の使用量の上限、配信がログに記録される方法、SMS の毎日の使用状況レポートをサブスクライブするかどうか、などを指定できます。

これらの設定は、アカウントから送信するすべての SMS メッセージに対して有効になりますが、個々のメッセージの送信時に上書きすることができます。詳細については、「携帯電話に発行する」を参照してください。

トピック

- を使用した SMS メッセージングプリファレンスの設定 AWS Management Console
- プリファレンスの設定 (AWS SDKs

を使用した SMS メッセージングプリファレンスの設定 AWS Management Console

1. Amazon SNS コンソールにサインインします。
2. SMS メッセージングをサポートしているリージョンを選択します。
3. ナビゲーションパネルで、[モバイル]、[(テキストメッセージング (SMS)] を選択します。
4. [モバイルテキストメッセージング (SMS)] ページの [テキストメッセージプリファレンス] セクションで、[編集] を選択します。
5. [テキストメッセージングの優先設定の編集] ページの [詳細] セクションで、以下の操作を実行します。

   a. [デフォルトメッセージタイプ] で、次のいずれかを選択します。

      - [プロモーション] — 重要度が低いメッセージ (広告など)。Amazon SNS は、最も低いコストが発生するようにメッセージ配信を最適化します。
      - [トランザクション] - 多要素認証のワンタイムパスコードなど、お客様のトランザクションをサポートするクリティカルメッセージ。Amazon SNS は、メッセージ配信を最も高い信頼性の実現のために最適化します。

      プロモーションおよびトランザクションメッセージの料金表の詳細については、「グローバル SMS 料金表」を参照してください。

   b. [アカウントの使用料制限] に、各暦月で SMS メッセージに使用する額 (USD) を入力します。

> **Important**
>
> - デフォルトでは、使用限度は 1.00 USD に設定されます。サービス限度を引き上げるには、リクエストを送信します。
> - コンソールに設定した量がサービス限度を超える場合、Amazon SNS は SMS メッセージの発行を停止します。
> - Amazon SNS は分散システムであるため、使用クォータを超えると数分以内に SMS メッセージの送信を停止します。その間に SMS メッセージを送信し続けると、クォータを超えるコストが発生する可能性があります。

6. (オプション) [デフォルトの送信者 ID] にカスタム ID (ビジネスブランドなど) を入力します。これは、受信デバイスに送信者として表示されます。

> **Note**
>
> 送信者 ID のサポートについては、国によって異なります。

7. (オプション) 使用状況レポート用の Amazon S3 バケット名の名前を入力します。

> **Note**
>
> S3 バケットポリシーによって Amazon SNS への書き込みアクセスが付与される必要があります。

8. [変更を保存] を選択します。

プリファレンスの設定 (AWS SDKs

AWS SDKs の 1 つを使用して SMS プリファレンスを設定するには、Amazon SNS API のSetSMSAttributesリクエストに対応するその SDK の アクションを使用します。 Amazon SNS このリクエストによって、月ごとの使用限度およびデフォルトの SMS タイプ (プロモーションまたはトランザクション) などの、異なる SMS の属性に値を割り当てます。利用可能な SMS 属性の詳細については、『Amazon Simple Notification Service API リファレンス』の「SetSMSAttributes」を参照してください。

以下のコード例は、SetSMSAttributes の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> の詳細については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon SNS を使用して defaultsMsType 属性を設定する方法。

```
//! 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();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「SetSMSAttributes」を参照してください。

CLI

AWS CLI

SMS メッセージ属性を設定するには

次の `set-sms-attributes` の例では、SMS メッセージのデフォルトの送信者 ID を `MyName` に設定します。

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「SetSMSAttributes」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> の詳細については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「SetSMSAttributes」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> の詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、AWS SDK for JavaScript API リファレンスの「SetSMSAttributes」を参照してください。

PHP

SDK for PHP

> **Note**
>
> の詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
$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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API リファレンスの「SetSMSAttributes」を参照してください。

SMS メッセージの送信

このセクションでは、SMS メッセージを送信する方法について説明します。

トピック

- トピックへ発行する
- 携帯電話に発行する

トピックへ発行する

Amazon SNS トピックにこれらの電話番号を登録することで、1 通の SMS メッセージを一度に複数の電話番号に送信できます。SNS トピックは通信チャンネルで、受信者を追加し、すべての受信者にメッセージを発行できます。サブスクリプションをキャンセルするか、受信者が AWS アカウントからの SMS メッセージの受信をオプトアウトするまで、トピックに発行されるすべてのメッセージを受け取ります。

トピック

- トピックへメッセージを送信する (コンソール)
- トピックへメッセージを送信する (AWS SDK)

トピックへメッセージを送信する (コンソール)

トピックを作成する

SMS メッセージを送信するトピックがまだない場合は、次の手順を実行します。

1. Amazon SNS コンソールにサインインします。

2. コンソールメニューで、AWS SMS メッセージングをサポートしているリージョン を選択します。
3. ナビゲーションペインで、[トピック] を選択します。
4. [トピック] ページで、[トピックの作成] を選択します。
5. [トピックの作成] ページの [詳細] で、次の操作を行います。

   a. [Type (タイプ)] で、[Standard (標準)] を選択します。
   b. [名前] にトピック名を入力します。
   c. (オプション) [表示名] に SMS メッセージのカスタムプレフィックスを入力します。トピックにメッセージを送信する場合、Amazon SNS によって右アングルブラケット (>) とスペースに続いて表示名が付加されます。表示名では大文字と小文字が区別されず、Amazon SNS は表示名を大文字に変換します。例えば、トピックの表示名が `MyTopic` で、メッセージが `Hello World!` である場合、メッセージは次のように表示されます。

   ```
   MYTOPIC> Hello World!
   ```

6. [Create topic] (トピックの作成) を選択します。トピックの名前と Amazon リソースネーム (ARN) が、[トピック] ページに表示されます。

SMS サブスクリプションを作成するには

トピックに 1 回だけメッセージを発行することによって、サブスクリプションを使用して SMS メッセージを複数の受信者に送信できます。

> **Note**
>
> Amazon SNS を使用して SMS メッセージを送信し始めると、AWS アカウントの所在地は SMS サンドボックス内になります。SMS サンドボックスは、SMS 送信者としての評判を損なうことなく、Amazon SNS 機能を試すための安全な環境を提供します。アカウントが SMS サンドボックスにあり、Amazon SNS のすべての機能を使用できますが、SMS メッセージを送信できるのは、確認済みの送信先の電話番号だけです。詳細については、「SMS サンドボックス」を参照してください。

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションペインで [サブスクリプション] を選択します。
3. [サブスクリプション] ページで [サブスクリプションの作成] を選択します。

4. [サブスクリプションの作成] ページで [詳細] セクションで、以下を実行します。

   a. トピック ARNで、SMS メッセージを送信するトピックの Amazon リソースネーム (ARN) を入力または選択します。

   b. [プロトコル] で SMS を選択します。

   c. [エンドポイント] に、トピックをサブスクライブする電話番号を入力します。

5. [サブスクリプションの作成] を選択します。サブスクリプション情報は [サブスクリプション] ページに表示されます。

   さらに電話番号を追加するには、このステップを繰り返します。電子メールなど、他の種類のサブスクリプションを追加することもできます。

メッセージを送信するには

トピックにメッセージを発行すると、Amazon SNS はトピックにサブスクライブされているすべての電話番号にメッセージの配信を試みます。

1. [Amazon SNS コンソール] の [トピック] ページで、SMS メッセージを送信するトピックの名前を選択します。
2. トピックの詳細ページで、[トピックへの発行] を選択します。
3. [トピックへのメッセージの発行] ページの [メッセージの詳細] で、次の操作を行います。

   a. トピックに E メールのサブスクリプションが含まれていない場合、および E メール と SMS のサブスクリプションの両方を発行する場合は、[件名] のフィールドを空のままにします。Amazon SNS の [件名] は、E メールの件名の欄とにあたります。

   b. (オプション) [有効期限 (TTL)] に、Amazon SNS がモバイルアプリケーションエンドポイント受信者に SMS メッセージを送信するのに必要な秒数を入力します。

4. [メッセージ本文] で、次の作業を行います。

   a. [メッセージ構造] で、[すべての配信プロトコルに同一のペイロード] をクリックして、トピックに登録されているすべてのプロトコルタイプに同じメッセージを送信します。または、[配信プロトコルごとにカスタムペイロード] を選択して、異なるプロトコルタイプの受信者用のメッセージをカスタマイズします。例えば、電話番号受信者にはデフォルトメッセージを入力し、E メール受信者にはカスタムメッセージを入力できます。

   b. [エンドポイントに送信するメッセージ本文] を選択し、メッセージを入力するか、配信プロトコルごとにカスタムメッセージを入力します。

トピックに表示名がある場合、Amazon SNS はそれをメッセージの長さを増やすメッセージに追加します。表示名の長さは、名前の文字数に Amazon SNS が追加する右アングルブラケット (>) とスペースの 2 文字をプラスしたものです。

SMS メッセージのサイズ限度の詳細については、「携帯電話に発行する」を参照してください。

5. (オプション) [メッセージ属性] で、タイムスタンプ、署名、ID などのメッセージメタデータを追加します。
6. [メッセージの発行] を選択します。Amazon SNS から SMS メッセージを送信し、成功のメッセージを表示します。

トピックへメッセージを送信する (AWS SDK)

AWS SDK を使用するには、認証情報を使用して設定する必要があります。詳細については、『AWS SDK とツールのリファレンスガイド』の「共有設定ファイルおよび認証情報ファイル」を参照してください。

次のコードの例は以下の方法を示しています。

- Amazon SNS トピックを作成します。
- 携帯電話番号をトピックにサブスクライブする。
- SMS メッセージをトピックに発行して、登録されているすべての電話番号がメッセージを一度に受信できるようにします。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

トピックを作成し、その ARN を返します。

```
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 "";
    }
}
```

トピックへのエンドポイントのサブスクライブ。

```
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);
        }
    }
}
```

送信者の ID、上限価格、タイプなど、メッセージ上の属性を設定します。メッセージ属性はオプションです。

```
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);
        }
    }
}
```

トピックへのメッセージの発行 メッセージは、すべての受信者に送信されます。

```
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);
        }
    }
}
```

携帯電話に発行する

Amazon SNS を使用して、Amazon SNS トピックに電話番号を登録せずに、SMS メッセージを携帯電話に直接送信できます。

> **Note**
>
> 1 つのメッセージを同時に複数の電話番号に送信する場合、電話番号をトピックにサブスクライブすると便利です。SMS メッセージをトピックに発行する方法については、「トピックへ発行する」を参照してください。

メッセージを送信するとき、メッセージがコストまたは信頼性に対して最適化されているかどうかを制御できます。送信者 ID または発信番号を指定することもできます。Amazon SNS API または AWS SDK を使用してメッセージをプログラムで送信する場合、メッセージ配信の上限価格を指定できます。

各 SMS メッセージは最大 140 バイトまで含めることができ、文字限度はエンコーディングスキームによって異なります。例えば、SMS メッセージには以下を含めることができます。

- 160 GSM 文字
- 140 ASCII 文字
- 70 UCS-2 文字

サイズ限度を超えてメッセージを発行する場合は、Amazon SNS により、複数のメッセージとして送信され、それぞれが文字数の限度以内に収められます。メッセージは単語の途中ではなく、単語の境目で切り離されます。1 回の SMS 発行アクションの合計サイズ限度は、1,600 バイトです。

SMS メッセージを送信するときは、E.164 形式を使用して電話番号を指定します。E.164 形式は国際電気通信に使用される標準電話番号付与構造です。この形式に従う電話番号は最大 15 桁を設定でき、プラス記号 (+) および国コードのプレフィックスが付いています。例えば、E.164 形式の米国の電話番号は +1XXX5550100 と表示されます。

トピック

- メッセージの送信 (コンソール)
- メッセージの送信 (SDK)AWS

メッセージの送信 (コンソール)

1. Amazon SNS コンソールにサインインします。
2. コンソールメニューで、AWS SMS メッセージングをサポートしているリージョン を選択します。
3. ナビゲーションペインで、[テキストメッセージング (SMS)] を選択します。
4. リポジトリの [モバイルテキストメッセージング (SMS)] ページで、[テキストメッセージの発行]。
5. リポジトリの [SMS メッセージを発行する] ページ、[メッセージの種類] で、次のいずれかを選択します。

- プロモーション - マーケティングメッセージなどの非クリティカルなメッセージ。
- [トランザクション] - 多要素認証のワンタイムパスコードなど、顧客のトランザクションをサポートするクリティカルメッセージ。

> **Note**
>
> このメッセージレベルの設定をすると、アカウントレベルのデフォルトメッセージタイプは上書きされます。アカウントレベルのデフォルトのメッセージタイプは、[モバイルテキストメッセージング (SMS)] ページの [テキストメッセージングの優先設定] のセクションで設定できます。

プロモーションおよびトランザクションメッセージの料金表の詳細については、「世界各地の SMS 料金表」を参照してください。

6. [送信先電話番号] に、メッセージを送信する電話番号を入力します。
7. [メッセージ] に、送信するメッセージを入力します。
8. (オプション) [送信元アイデンティティ] で、受信者を識別する方法を指定します。

- (オプション) [送信者 ID] に、少なくとも 1 つの文字を含み、スペースは含まない、3 ~ 11 文字の英数字のカスタム ID を入力します。送信者 ID は、受信側デバイスにメッセージ送信者として表示されます。例えば、メッセージソースを識別しやすいように、ビジネスブランドを使用できます。

  送信者 ID のサポートについては、国、リージョン、またはその両方によって異なります。例えば、米国の電話番号へ配信されるメッセージは、送信者 ID を表示しません。送信者 ID をサポートする国およびリージョンについては、「サポートされている国と地域」を参照してください。

  送信者 ID を指定しない場合、次のいずれか 1 つが送信元 ID として表示されます。

  - ロングコードをサポートしている国では、ロングコードが表示されます。
  - 送信者 ID のみがサポートされている国では、「NOTICE」が表示されます。

  このメッセージレベルの送信者 ID は、[テキストメッセージプリファレンス] ページで設定するデフォルトの送信者 ID を上書きします。

- [送信元番号] を指定するには、受信者のデバイス上で送信者の電話番号として表示する 5～14 の数字の文字列を入力します。この文字列は、送信先の国のお客様の AWS アカウント に設定される送信元番号と一致シている必要があります。送信元番号は、10DLC 番号、フリーダイヤル、ロングコード、ショートコードのいずれでもかまいません。 person-to-person 詳細については、「SMS メッセージの送信元アイデンティティ」を参照してください。

  発信元番号を指定しない場合、Amazon SNS は設定に基づいて SMS テキストメッセージに使用する発信番号を選択します。 AWS アカウント

9. インドの受取人に SMS メッセージを送信する場合は、[国固有の属性 を展開し、次の属性を指定します。

   - エンティティ ID - インドの受信者に SMS メッセージを送信するためのエンティティ ID または プリンシパルエンティティ (PE) ID。この ID は、1 ～ 50 文字の一意の文字列で、Telecom Regulatory Authority of India (TRAI) が提供する、TRAI に登録したエンティティを識別するものです。
   - テンプレート ID - インドの受信者に SMS メッセージを送信するためのテンプレート ID。この ID は、TRAI に登録したテンプレートを識別する 1 ～ 50 文字の一意の文字列です。テンプレート ID は、メッセージに対して指定した送信者 ID に関連付ける必要があります。

   インドの受信者に SMS メッセージを送信する方法の詳細については、「インドの送信者 ID 登録要件」を参照してください。

10. [メッセージの発行] を選択します。

> **Tip**
>
> 送信元番号から SMS メッセージを送信するには、Amazon SNS コンソールのナビゲーションパネルで送信元番号を送信します。[機能] 列で [SMS] 含む送信元番号を選択し、次に [テキストメッセージの発行] を選択します。

メッセージの送信 (SDK)AWS

いずれかの AWS SDK を使用して SMS メッセージを送信するには、Amazon SNS API `Publish` のリクエストに対応するその SDK の API オペレーションを使用します。このリクエストでは、電話番号に SMS メッセージを直接送信できます。次の属性名の値を設定する場合、`MessageAttributes` パラメータも使用できます。

`AWS.SNS.SMS.SenderID`

少なくとも 1 つの文字を含み、スペースは含まない、3 ~ 11 文字の英数字のカスタム ID。送信者 ID は、受信デバイスにメッセージ送信者として表示されます。例えば、メッセージ ソースを識別しやすいように、ビジネスブランドを使用できます。

送信者 ID のサポートについては、国またはリージョンによって異なります。例えば、米国の電話番号へ配信されるメッセージは、送信者 ID を表示しません。送信者 ID をサポートする国およびリージョンのリストについては、「サポートされている国と地域」を参照してください。

送信者 ID を指定しない場合、サポートされている国またはリージョンでは、メッセージは送信者 ID としてロングコードを表示します。アルファベットの送信者 ID を必要とする国またはリージョンでは、送信者 ID は「NOTICE」と表示されます。

このメッセージレベルの属性は、`SetSMSAttributes` リクエストを使用して設定する、アカウントレベルの属性 `DefaultSenderID` を上書きします。

`AWS.MM.SMS.OriginationNumber`

5 ~ 14 個の数値のカスタム文字列で、オプションの先頭にプラス記号 (+) を付けられます。この番号文字列は、受信側のデバイスの送信者の電話番号として表示されます。文字列は、 AWS 送信先の国のアカウントで設定されている送信元番号と一致する必要があります。発信元番号は、10DLC 番号、フリーダイヤル、 person-to-person (P2P) ロングコード、ショートコードのいずれでもかまいません。詳細については、「送信元番号」を参照してください。

オリジネーション番号を指定しない場合、Amazon SNS はアカウント設定に基づいてオリジネーション番号を選択します。 AWS

`AWS.SNS.SMS.MaxPrice`

SMS メッセージの送信に使用できる上限価格 (USD)。Amazon SNS がメッセージの送信により上限価格を超えるコストが発生すると判断した場合、Amazon SNS はメッセージを送信しません。

month-to-date SMS コストがその属性に設定されたクォータをすでに超えている場合、この属性は効果がありません。`MonthlySpendLimitSetSMSAttributes` リクエストを使用して `MonthlySpendLimit` 属性を設定できます。

Amazon SNS トピックにメッセージを送信する場合、トピックにサブスクライブされている各電話番号への各メッセージの配信に上限価格が適用されます。

`AWS.SNS.SMS.SMSType`

送信するメッセージのタイプ。

- `Promotional` (デフォルト) - マーケティングメッセージなどの非クリティカルなメッセージ。
- `Transactional` - 多要素認証のワンタイムパスコードなど、顧客のトランザクションをサポートするクリティカルメッセージ。

このメッセージレベルの属性は、`SetSMSAttributes` リクエストを使用して設定する、アカウントレベルの属性 `DefaultSMSType` を上書きします。

`AWS.MM.SMS.EntityId`

この属性は、インドの受信者に SMS メッセージを送信するときにのみ必要です。

これは、インドの受信者に SMS メッセージを送信するためのエンティティ ID またはプリンシパルエンティティ (PE) ID です。この ID は、1 ~ 50 文字の一意の文字列で、Telecom Regulatory Authority of India (TRAI) が提供する、TRAI に登録したエンティティを識別するものです。

`AWS.MM.SMS.TemplateId`

この属性は、インドの受信者に SMS メッセージを送信するときにのみ必要です。

これは、インドの受信者に SMS メッセージを送信するためのテンプレートです。この ID は、TRAI に登録したテンプレートを識別する 1 ~ 50 文字の一意の文字列です。テンプレート ID は、メッセージに対して指定した送信者 ID に関連付ける必要があります。

メッセージを送信する

次のコード例は、Amazon SNS を使用して SMS メッセージを発行する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> まだまだあります。 GitHubAWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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}");
            }
        }
    }
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「発行」を参照してください。

C++

SDK for C++

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * 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();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「発行」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「Publish」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Publish」を参照してください。

PHP

SDK for PHP

> **Note**
>
> まだまだあります GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API リファレンスの「発行」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「Publish」を参照してください。

SMS のアクティビティをモニタリングする

SMS をモニタリングすることによって、送信先の電話番号、正常または失敗した配信、失敗の原因、コスト、およびその他の情報を追跡できます。Amazon SNS は、コンソールで統計をまとめ、Amazon CloudWatch に情報を送信し、指定した Amazon S3 バケットに SMS の毎日の使用状況レポートを送信します。

トピック

- SMS 配信統計を表示する
- SMS 配信用の Amazon CloudWatch のメトリクスおよびログを表示する
- SMS の毎日の使用状況レポートを表示する

SMS 配信統計を表示する

最新の SMS 配信に関する統計を表示するために Amazon SNS コンソールを使用できます。

1. Amazon SNS コンソールにサインインします。
2. コンソールメニューで、SMS メッセージングをサポートしているリージョンにリージョンセレクタを設定します。
3. ナビゲーションパネルで、[テキストメッセージング (SMS)] を選択します。
4. [テキストメッセージ (SMS)] ページの、[Account stats] セクションで、トランザクションおよびプロモーション SMS メッセージ配信のグラフを表示します。各グラフは、前述の 15 日間の次のデータを表示します。

   - 配信成功率 (正常な配信の割合)
   - 送信 (配信試行数)
   - 失敗 (配信の失敗数)

このページで、[使用状況]ボタンを選択して、毎日の使用状況レポートを保存する Amazon S3 バケットに移動することもできます。詳細については、「SMS の毎日の使用状況レポートを表示する」を参照してください。

SMS 配信用の Amazon CloudWatch のメトリクスおよびログを表示する

Amazon CloudWatch および Amazon CloudWatch Logs を使用して、SMS のメッセージ配信をモニタリングできます。

トピック

- Amazon CloudWatch メトリクスを表示する
- CloudWatch Logs を表示する
- 正常な SMS 配信のログの例
- 失敗した SMS 配信のログの例
- SMS 配信の失敗の原因

Amazon CloudWatch メトリクスを表示する

Amazon SNS は、自動的に SMS メッセージ配信に関するメトリクスを収集し、これらを Amazon CloudWatch にプッシュします。CloudWatch を使用してこれらのメトリクスをモニタリングし、そのメトリクスがしきい値を超えたときに警告するようにアラームを作成できます。例えば、CloudWatch メトリクスをモニタリングして、SMS 配信率と今月の始めから今日までの SMS 料金を知ることができます。

CloudWatch メトリクスのモニタリング、 アラームの設定、利用可能なメトリクスの種類の詳細については、「CloudWatch を使用した Amazon SNS のモニタリング」を参照してください。

CloudWatch Logs を表示する

Amazon SNS による Amazon CloudWatch Logs への書き込みを有効化することにより、正常および失敗した SMS メッセージ配信の情報を収集できます。送信した各 SMS メッセージに対して、Amazon SNS は、メッセージの価格、成功または失敗のステータス、失敗の理由 (メッセージが失敗した場合)、メッセージのドウェル時間、その他の情報を含むログを書き込みます。

SMS メッセージに対して CloudWatch Logs を有効にして表示するには

1. Amazon SNS コンソールにサインインします。
2. コンソールメニューで、SMS メッセージングをサポートしているリージョンにリージョンセレクタを設定します。

3. ナビゲーションパネルで、[テキストメッセージング (SMS)] を選択します。
4. [モバイルテキストメッセージング (SMS)] ページの [テキストメッセージングの優先設定] セクションで、[編集] を選択します。
5. 次のページで、[Delivery status logging] セクションに追加します。
6. [成功サンプルの割合] で、Amazon SNS が CloudWatch Logs でログを書き込む正常な SMS 配信の割合を指定します。例:

   - 失敗した配信のみをログを書き込むには、この値を 0 に設定します。
   - 正常な配信の 10% に対してログを書き込むには、10 に設定します。

   割合を指定しないなら、Amazon SNS は、すべての正常配信に対してログを書き込みます。
7. 必要なアクセス権限を提供するには、次のいずれかを行います。

   - 新しいサービスロールを作成するには、[新しいサービスロールの作成]を選択 し、[新しいロールの作成] を選択します。次のページで、[許可] を使用して、Amazon SNS にアカウントのリソースへの書き込みアクセス許可を付与します。
   - 既存のサービスロールを使用するには、[既存のサービスロールを使用する] を選択し、ARN 名を [成功した配信および失敗した配信用の IAM ロール] ボックスにペーストします。

     指定するサービスロールは、アカウントのリソースへの書き込みアクセスを許可する必要があります。IAM ロールの作成の詳細については、『IAM ユーザーガイド』の「AWS のサービス用ロールの作成 (コンソール)」を参照してください。
8. [Save changes] (変更の保存) をクリックします。
9. [モバイルテキストメッセージング (SMS)] ページに戻り、[配信ステータスログ] セクションに移動し、使用可能なログを表示します。

> **Note**
>
> 送信先の電話番号のキャリアによっては、Amazon SNS コンソールに配信ログが表示されるまでに最大 72 時間かかる場合があります。

正常な SMS 配信のログの例

正常な SMS 配信の配信ステータスのログは次の例のようになります。

```
{
```

```
    "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"
}
```

失敗した SMS 配信のログの例

失敗した SMS 配信の配信ステータスのログは次の例のようになります。

```
{
    "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 配信の失敗の原因

失敗の理由は、`providerResponse` 属性で提供されます。SMS メッセージは以下の理由で送信に失敗することがあります。

- 電話業者がスパムとしてブロック
- 送信先がブロックリストにある
- 無効な電話番号
- メッセージ本文が無効
- 電話業者がこのメッセージをブロック
- 電話業者が現在、到達不可能/使用不可能
- 電話が SMS をブロック
- 電話がブロックリストにある
- 電話が現在、到達不可能/使用不可能
- 電話番号はオプトアウトしている
- この配信が上限価格を超えている
- 電話に達する際の原因不明なエラー

SMS の毎日の使用状況レポートを表示する

Amazon SNS から毎日の使用状況レポートにサブスクライブすることによって、SMS 配信をモニタリングできます。1 つ以上の SMS メッセージを送信する日ごとに、Amazon SNS は、指定した Amazon S3 バケットに CSV ファイルとして使用状況レポートを配信します。SMS 使用状況レポートが S3 バケットで利用可能になるまでに、24 時間かかります。

トピック

- 毎日の使用状況レポートの情報
- 毎日の使用状況レポートにサブスクライブする

毎日の使用状況レポートの情報

使用状況レポートには、アカウントから正常に配信した各 SMS メッセージに関する以下の情報が含まれています。

このレポートには、オプトアウトした受取人に送信されるメッセージは含まれない点に注意してください。

- メッセージの発行時間 (UTC)
- メッセージ ID
- 送信先の電話番号
- メッセージの種類
- 配信ステータス
- メッセージの価格 (USD)
- パート番号 (1 通のメッセージとして長すぎる場合、メッセージは複数のパートに分割されます)
- パートの総数

> **Note**
>
> Amazon SNS がパート番号を受け取らなかった場合は、その値をゼロに設定します。

毎日の使用状況レポートにサブスクライブする

毎日の使用状況レポートにサブスクライブするには、適切なアクセス権限を持つ Amazon S3 バケットを作成する必要があります。

毎日の使用状況レポート用の Amazon S3 バケットを作成するには

1. SMS メッセージを送信する AWS アカウント から、Amazon S3 コンソールにサインインします。
2. [バケットの作成] を選択します。
3. [バケット名] には、アカウントと組織で一意の名前を入力することをお勧めします。例えば、パターン `<my-bucket-prefix>-<account_id>-<org-id>` を使用します。

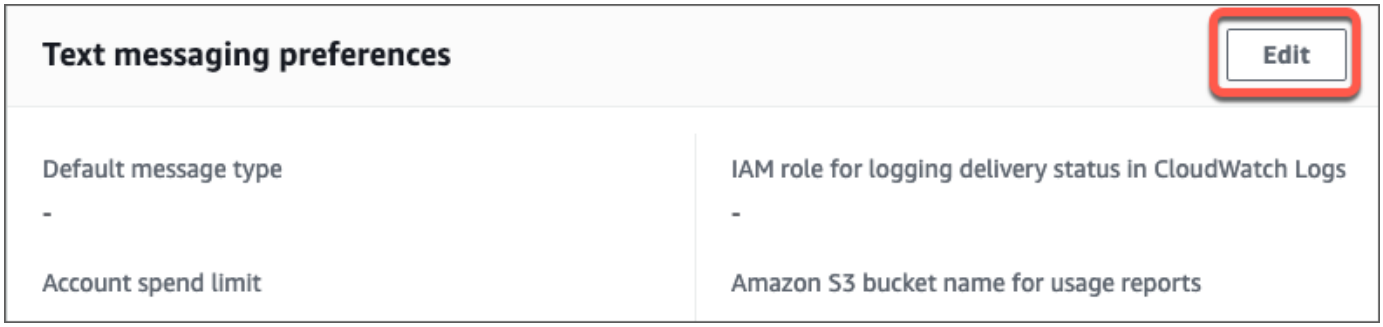
   バケット名に関する規則と制限については、『Amazon Simple Storage Service ユーザーガイド』の「バケットの命名規則」を参照してください。
4. [Create] を選択します。
5. [すべてのバケット]テーブルでバケットを選択します。
6. [Permissions] セクションで、[バケットポリシー] を選択します。
7. [バケットポリシーエディター] ウィンドウで、Amazon SNS サービスプリンシパルをバケットに書き込むことを許可するポリシーを指定します。例については、バケットポリシーの例 を参照してください。

サンプルポリシーを使用する場合は、*my-s3-bucket* をステップ 3 で選択したバケット名に置き換えてください。

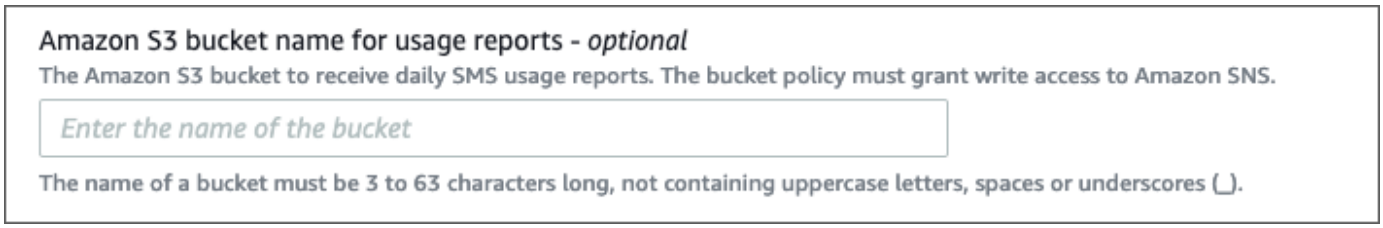
8. [Save] を選択します。

毎日の使用状況レポートにサブスクライブするには

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[テキストメッセージング (SMS)] を選択します。
3. [テキストメッセージング (SMS)] ページの [テキストメッセージングの設定] セクションで、[編集] を選択します。



4. [テキストメッセージングの設定の編集] ページの [詳細] セクションで、[使用状況レポートに使用する Amazon S3 バケット名] を指定します。



5. [Save changes] (変更の保存) をクリックします。

バケットポリシーの例

以下のポリシーは、Amazon SNS サービスプリンシパルに`s3:PutObject`、`s3:GetBucketLocation` および `s3:ListBucket` アクションの実行を許可します。

AWS は、お客様のアカウントのリソースへのアクセスを与えられたサービスプリンシパルを持つすべてのサービスを目的としたツールを提供します。Amazon S3 バケットポリシーステートメントのプリンシパルが AWS service principal (AWS のサービスプリンシパル) で

ある場合、aws:SourceArn または aws:SourceAccount のグローバル条件キーを使用して、confused deputy problem (混乱した代理の問題) から保護することができます。バケットが毎日の使用状況レポートを受信できるリージョンとアカウントを制限するには、以下の例のように aws:SourceArn を使用します。これらのレポートを生成できるリージョンを制限したくない場合は、aws:SourceAccount を使用して、レポートを生成しているアカウントに基づいて制限します。リソースの ARN が不明の場合は、aws:SourceAccount を使用してください。

Amazon S3 バケットを作成して、Amazon SNS から毎日の SMS 使用状況レポートを受信する際に、混乱した代理保護を含む次の例を使用します。

```
{
 "Version": "2008-10-17",
 "Statement": [{
   "Sid": "AllowPutObject",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::my-s3-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:::my-s3-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:::my-s3-bucket",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "account_id"
     },
     "ArnLike": {
      "aws:SourceArn": "arn:aws:sns:region:account_id:*"
     }
    }
   }
  ]
}
```

> **Note**
>
> Amazon S3 ポリシーの Condition エレメントで指定される AWS アカウント に所有される Amazon S3 バケットに使用状況レポートを発行できます。他の AWS アカウント が所有する Amazon S3 バケットに使用状況レポートを発行するには、「How can I copy S3 objects from another AWS アカウント?」を参照してください。

毎日の使用状況レポートの例

毎日の使用状況レポートにサブスクライブすると、Amazon SNS は毎日、以下の場所に使用状況データを使用して CSV ファイルを配置します。

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

各ファイルには、最大 50,000 のレコードを含めることができます。1 日のレコードがこの限度を超えると、Amazon SNS は複数のファイルを追加します。

レポートの例を以下に示します。

```
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
```

電話番号および SMS サブスクリプションを管理する

Amazon SNS には、アカウントから SMS メッセージを受信するユーザーを管理するためのオプションが複数用意されています。制限された頻度で、アカウントからの SMS メッセージの受信をオプトアウトした電話番号をオプトインできます。SMS サブスクリプションへのメッセージの送信を停止するために、それらに発行されるサブスクリプションやトピックを削除できます。

トピック

- SMS メッセージの受信をオプトアウトする
- 電話番号とサブスクリプション (コンソール) を管理する
- 電話番号およびサブスクリプション (AWS SDK) を管理する

SMS メッセージの受信をオプトアウトする

現地の法律および規制により義務付けられている場合 (米国およびカナダなど)、SMS の受信者は、デバイスを利用して以下のいずれかのメッセージに返信することによって、オプトアウトできます。

- ARRET (フランス語)
- CANCEL
- END
- OPT-OUT
- OPTOUT
- QUIT
- REMOVE

- STOP
- TD
- UNSUBSCRIBE

オプトアウトするには、受信者はAmazon SNS がメッセージの配信に使用したものと同じ送信元番号に返信する必要があります。オプトアウトすると、電話番号をオプトイン AWS アカウント しない限り、受信者は から配信された SMS メッセージを受信しなくなります。

電話番号が Amazon SNS トピックにサブスクライブされると、オプトアウトはサブスクリプションを削除しませんが、電話番号をオプトインしない限り、SMS メッセージは、サブスクリプションへ送信できません。

電話番号とサブスクリプション (コンソール) を管理する

Amazon SNS コンソールを使用して、アカウントで SMS メッセージを受信する電話番号を管理できます。

オプトアウトされた電話番号をオプトインする

ユーザーのアカウントからの SMS メッセージの受信をオプトアウトした電話番号を表示し、これらの電話番号をオプトインしてメッセージの送信を再開することができます。

30 日に 1 回のみ、電話番号をオプトインできます。

1. Amazon SNS コンソールにサインインします。
2. コンソールメニューで、SMS メッセージングをサポートしているリージョンにリージョンセレクタを設定します。
3. ナビゲーションパネルで、[テキストメッセージング (SMS)] を選択します。
4. [テキストメッセージング (SMS)] ページで、[オプトアウトされた電話番号を表示] を選択します。[Opted out phone numbers] ページは、オプトアウトされた電話番号を表示します。
5. オプトインする電話番号のチェックボックスをオンにして、[オプトイン] を選択します。電話番号はオプトアウトではなくなり、送信した SMS メッセージを受信します。

SMS サブスクリプションを削除する

トピックにパブリッシュするときに、その電話番号への SMS メッセージの送信を停止するように SMS サブスクリプションを削除します。

1. ナビゲーションパネルで、[サブスクリプション] を選択します。
2. 削除するサブスクリプションのチェックボックスをオンにします。それから、[アクション] を選択して、[アプリケーションの削除] を選択します。
3. [削除] ウィンドウで、[削除] を選択します。Amazon SNS はサブスクリプションを削除し、成功のメッセージを表示します。

トピックを削除する

サブスクライブしたエンドポイントにメッセージを発行しない場合トピックを削除します。

1. ナビゲーションパネルで、[トピック] を選択します。
2. 削除するトピックのチェックボックスをオンにします。[アクション] を選択してから、[トピックの削除] を選択します。
3. [削除] ウィンドウで、[削除] を選択します。Amazon SNS はトピックを削除し、成功のメッセージを表示します。

電話番号およびサブスクリプション (AWS SDK) を管理する

AWS SDKs を使用して Amazon SNS にプログラムでリクエストを行い、アカウントから SMS メッセージを受信できる電話番号を管理できます。

AWS SDK を使用するには、 認証情報を使用して設定する必要があります。詳細については、「AWS SDK とツールのリファレンスガイド」の「共有設定ファイルおよび認証情報ファイル」を参照してください。

すべてのオプトアウトした電話番号を表示する

すべてのオプトアウトした電話番号を表示するには、Amazon SNS API を使用して、`ListPhoneNumbersOptedOut` リクエストを送信します。

以下のコード例は、`ListPhoneNumbersOptedOut` の使用方法を示しています。

CLI

AWS CLI

SMS メッセージのオプトアウトを一覧表示するには

次の `list-phone-numbers-opted-out` の例では、SMS メッセージの受信をオプトアウトした電話番号を一覧表示しています。

```
aws sns list-phone-numbers-opted-out
```

出力:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- API の詳細については、「 コマンドリファレンスListPhoneNumbersOptedOut」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスListPhoneNumbersOptedOut」の「」を参照してください。 AWS SDK for Java 2.x

PHP

SDK for PHP

> **Note**
>
> には他にもがあります GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスListPhoneNumbersOptedOut」の「」を参照してください。 AWS SDK for PHP

電話番号がオプトアウトしているかどうかを確認する

電話番号がオプトアウトしているかどうかを確認するには、Amazon SNS API を使用して、`CheckIfPhoneNumberIsOptedOut` リクエストを送信します。

以下のコード例は、`CheckIfPhoneNumberIsOptedOut` の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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}");
            }
        }
    }
```

- API の詳細については、「 API リファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

電話番号での SMS メッセージのオプトアウトを確認するには

次のcheck-if-phone-number-is-opted-out例では、指定した電話番号が現在の AWS アカウントからの SMS メッセージの受信をオプトアウトしているかどうかを確認します。

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

出力:

```
{
    "isOptedOut": false
}
```

- API の詳細については、「 コマンドリファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> には他にもがあります GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS SDK for JavaScript

PHP

SDK for PHP

> **Note**
>
> には他にもがあります GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS SDK for PHP

オプトアウトされた電話番号をオプトインする

電話番号をオプトインするには、Amazon SNS API を使用して、`OptInPhoneNumber` リクエストを送信します。

30 日に 1 回のみ、電話番号をオプトインできます。

SMS サブスクリプションを削除する

Amazon SNS トピックからの SMS サブスクリプションを削除するには、Amazon SNS API を使用して `ListSubscriptions` リクエストを送信することにより、サブスクリプション ARN を取得し、次に、その ARN を `Unsubscribe` リクエストに渡します。

以下のコード例は、`Unsubscribe` の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

サブスクリプション ARN でトピックからサブスクライブを解除します。

```
/// <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;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「Unsubscribe」を参照してください。

C++

SDK for C++

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「Unsubscribe」を参照してください。

CLI

AWS CLI

トピックからサブスクライブを解除するには

次の unsubscribe の例では、指定したサブスクリプションをトピックから削除します。

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「Unsubscribe」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「Unsubscribe」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> には他にもがあります GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、AWS SDK for JavaScript API リファレンスの「Unsubscribe」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Unsubscribe」を参照してください。

PHP

SDK for PHP

> **Note**
>
> には他にもがあります GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API リファレンスの「Unsubscribe」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「Unsubscribe」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの「Unsubscribe」(サブスクリプション解除) を参照してください。

トピックを削除する

トピックとサブスクリプションすべてを削除するには、Amazon SNS API を使用して `ListTopics` リクエストを送信することでトピック ARN を取得し、次に、その ARN を `DeleteTopic` リクエストに渡します。

以下のコード例は、`DeleteTopic` の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピック ARN でトピックを削除します。

```
/// <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;
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

SNS トピックを削除するには

次の delete-topic の例では、指定した SNS トピックを削除します。

```
aws sns delete-topic \
```

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

このコマンドでは何も出力されません。

- API の詳細については、「 コマンドリファレンスDeleteTopic」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for Go

Java

SDK for Java 2.x

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> には他にもがあります GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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
  //   }
  // }
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、DeleteTopicAWS 「 SDK for Kotlin API リファレンス」の「」を参照してください。

PHP

SDK for PHP

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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());
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。AWS SDK for PHP

Python

SDK for Python (Boto3)

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、DeleteTopicAWS 「 SDK for Python (Boto3) API リファレンス」の「」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> には他にもがあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、DeleteTopicAWS 「 SDK for SAP ABAP API リファレンス」の「」を参照してください。

サポートされている国と地域

> **Important**
>
> 2023 年 8 月 31 日より、米国および米国領 (グアム、プエルトリコ、米領サモア諸島、および US ヴァージン諸島)に SMS メッセージを送信する際は、10DLC 番号または通話無料番号などの専用番号が必要となります。

現在、Amazon SNS AWS は次のリージョンで SMS メッセージングをサポートしています。

リージョン名	リージョン	エンドポイント	プロトコル
米国東部 (オハイオ)	us-east-2	sns.us-east-2.amazonaws.com	HTTP および HTTPS
米国東部 (バージニア北部)	us-east-1	sns.us-east-1.amazonaws.com	HTTP および HTTPS
米国西部 (北カリフォルニア)	us-west-1	sns.us-west-1.amazonaws.com	HTTP および HTTPS
米国西部 (オレゴン)	us-west-2	sns.us-west-2.amazonaws.com	HTTP および HTTPS
アフリカ (ケープタウン)	af-south-1	sns.af-south-1.amazonaws.com	HTTP および HTTPS
アジアパシフィック (ハイデラバード)	ap-south-2	sns.ap-south-2.amazonaws.com	HTTP および HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
アジアパシフィック (ジャカルタ)	ap-southeast-3	sns.ap-southeast-3.amazonaws.com	HTTP および HTTPS
アジアパシフィック (メルボルン)	ap-southeast-4	sns.ap-southeast-4.amazonaws.com	HTTP および HTTPS
アジアパシフィック (ムンバイ)	ap-south-1	sns.ap-south-1.amazonaws.com	HTTP および HTTPS
アジアパシフィック (大阪)	ap-northeast-3	sns.ap-northeast-3.amazonaws.com	HTTP および HTTPS
アジアパシフィック (シンガポール)	ap-southeast-1	sns.ap-southeast-1.amazonaws.com	HTTP および HTTPS
アジアパシフィック (シドニー)	ap-southeast-2	sns.ap-southeast-2.amazonaws.com	HTTP および HTTPS
アジアパシフィック (東京)	ap-northeast-1	sns.ap-northeast-1.amazonaws.com	HTTP および HTTPS
カナダ (中部)	ca-central-1	sns.ca-central-1.amazonaws.com	HTTP および HTTPS
欧州 (フランクフルト)	eu-central-1	sns.eu-central-1.amazonaws.com	HTTP および HTTPS
欧州 (アイルランド)	eu-west-1	sns.eu-west-1.amazonaws.com	HTTP および HTTPS
欧州 (ロンドン)	eu-west-2	sns.eu-west-2.amazonaws.com	HTTP および HTTPS
ヨーロッパ (ミラノ)	eu-south-1	sns.eu-south-1.amazonaws.com	HTTP および HTTPS
欧州 (パリ)	eu-west-3	sns.eu-west-3.amazonaws.com	HTTP および HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
欧州 (スペイン)	eu-south-2	sns.eu-south-2.amazonaws.com	HTTP および HTTPS
欧州 (ストックホルム)	eu-north-1	sns.eu-north-1.amazonaws.com	HTTP および HTTPS
欧州 (チューリッヒ)	eu-central-2	sns.eu-central-2.amazonaws.com	HTTP および HTTPS
イスラエル (テルアビブ)	il-central-1	sns.il-central-1.amazonaws.com	HTTP および HTTPS
中東 (バーレーン)	me-south-1	sns.me-south-1.amazonaws.com	HTTP および HTTPS
中東 (アラブ首長国連邦)	me-central-1	sns.me-central-1.amazonaws.com	HTTP および HTTPS
南米 (サンパウロ)	sa-east-1	sns.sa-east-1.amazonaws.com	HTTP および HTTPS
AWS GovCloud (米国東部)	us-gov-east-1	SNS。 us-gov-east-1.amazonaws.com	HTTP および HTTPS
AWS GovCloud (米国西部)	us-gov-west-1	SNS。 us-gov-west-1.amazonaws.com	HTTP および HTTPS

Amazon SNS を使用して、以下の国とリージョンに SMS メッセージを送信できます。

> **Note**
>
> サポートされている国で送信者 ID を使用すると、SMS の配信が向上します。

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
A						
アフガニスタン	AF	93	いいえ	いいえ	はい	いいえ
アルバニア	AL	355	いいえ	いいえ	はい	いいえ
アルジェリア	DZ	213	いいえ	いいえ	はい	いいえ
アンドラ	AD	376	いいえ	いいえ	はい	いいえ
アンギラ	AI	1-264	いいえ	いいえ	はい	いいえ
アンティグアバーブーダ	AG	1-268	いいえ	いいえ	はい	いいえ
アルゼンチン	AR	54	はい	いいえ	いいえ	[いいえ]
アルメニア	AM	374	いいえ	いいえ	はい	いいえ
アルバ	AW	297	いいえ	いいえ	はい	いいえ
オーストラリア	AU	61	いいえ	はい	要登録 [1]	はい
オーストリア	AT	43	はい	はい	はい	はい
アゼルバイジャン	AZ	994	いいえ	いいえ	はい	いいえ
B						

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
バハマ諸島	BS	1-242	いいえ	いいえ	いいえ	[いいえ]
バーレーン	BH	973	いいえ	いいえ	はい	いいえ
バングラデシュ	BD	880	いいえ	いいえ	はい	いいえ
バルバドス	BB	1-246	いいえ	いいえ	はい	いいえ
ベラルーシ	BY	375	いいえ	[いいえ]	要登録 [1]	[いいえ]
ベルギー	BE	32	[いいえ]	はい	いいえ	はい
ベリーズ	BZ	501	いいえ	いいえ	はい	いいえ
バミューダ	BM	1-441	いいえ	いいえ	はい	いいえ
ブータン	BT	975	いいえ	いいえ	はい	いいえ
ボリビア	BO	591	いいえ	いいえ	はい	いいえ
ボスニアヘルツェゴビナ	BA	387	いいえ	いいえ	はい	いいえ
ボツワナ	BW	267	いいえ	いいえ	はい	いいえ
ブラジル	BR	55	はい	いいえ	いいえ	はい
ブルネイ	BN	673	いいえ	いいえ	はい	いいえ
ブルガリア	BG	359	はい	いいえ	はい	はい
ブルキナファソ	BF	226	いいえ	いいえ	はい	いいえ
ブルンジ	BL	257	いいえ	いいえ	はい	いいえ

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
C						
カンボジア	KH	855	いいえ	いいえ	はい	いいえ
カメルーン	CM	237	いいえ	いいえ	はい	いいえ
カナダ	CA	1	はい	はい	いいえ	はい
カーボベルデ	CV	238	いいえ	いいえ	はい	いいえ
ケイマン諸島	KY	1-345	いいえ	いいえ	いいえ	[いいえ]
中央アフリカ共和国	CF	236	いいえ	いいえ	はい	いいえ
チャド	TD	235	いいえ	いいえ	はい	いいえ
チリ	CL	56	はい	はい	いいえ	はい
中国	CN	86	はい	いいえ	なし [2]	はい
コロンビア	CO	57	はい	はい	いいえ	はい
コモロ	KM	269	いいえ	いいえ	はい	いいえ
クック諸島	CK	682	いいえ	いいえ	はい	はい
コスタリカ	CR	506	いいえ	いいえ	いいえ	[いいえ]
クロアチア	HR	385	いいえ	いいえ	はい	いいえ
キプロス	CY	357	いいえ	いいえ	はい	いいえ
チェコ共和国	CZ	420	[いいえ]	はい	はい	はい

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
D						
コンゴ民主共和国	CD	243	いいえ	いいえ	はい	いいえ
デンマーク	DK	45	はい	はい	はい	はい
ジブチ	DJ	253	いいえ	いいえ	はい	いいえ
ドミニカ	DM	1-767	いいえ	いいえ	はい	いいえ
ドミニカ共和国	DO	1-809、1-829、1-849	はい	いいえ	いいえ	はい
E						
エクアドル	EC	593	はい	いいえ	いいえ	はい
エジプト	EG	20	はい	いいえ	要登録 [1]	はい
エルサルバドル	SV	503	いいえ	いいえ	いいえ	[いいえ]
赤道ギニア	GQ	240	いいえ	いいえ	はい	いいえ
エリトリア	ER	291	いいえ	いいえ	はい	いいえ
エストニア	EE	372	[いいえ]	はい	はい	はい
エチオピア	ET	251	いいえ	いいえ	はい	いいえ
F						
フェロー諸島	FO	298	いいえ	いいえ	はい	いいえ
フィジー	FJ	679	いいえ	いいえ	はい	いいえ

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
フィンランド	FI	358	はい	はい	はい	はい
フランス	FR	33	はい	いいえ	はい	はい
フランス領ギアナ	GF	594	いいえ	いいえ	はい	いいえ
フランス領ポリネシア	PF	689	いいえ	いいえ	はい	いいえ
G						
ガボン	GA	241	いいえ	いいえ	はい	いいえ
ガンビア	GM	220	いいえ	いいえ	はい	いいえ
ジョージア	GE	995	いいえ	いいえ	はい	いいえ
ドイツ	DE	49	はい	はい	はい	はい
ガーナ	GH	233	いいえ	いいえ	はい	いいえ
ジブラルタル	GI	350	いいえ	いいえ	はい	いいえ
ギリシャ	GR	30	いいえ	いいえ	はい	いいえ
グリーンランド	GL	299	いいえ	いいえ	はい	いいえ
グレナダ	GD	1-473	いいえ	いいえ	はい	いいえ
グアドループ	GP	590	いいえ	いいえ	はい	いいえ
グアム	GU	1-671	いいえ	いいえ	いいえ	はい

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
グアテマラ	GT	502	いいえ	いいえ	いいえ	[いいえ]
ガーンジー代官管轄区	GG	44-1481	いいえ	いいえ	はい	いいえ
ギニア	GN	224	いいえ	いいえ	はい	いいえ
ギニアビサウ	GW	245	いいえ	いいえ	はい	該当なし
ガイアナ	GY	592	いいえ	いいえ	はい	いいえ
H						
ハイチ	H	509	いいえ	いいえ	はい	いいえ
ホンジュラス	HN	504	いいえ	いいえ	はい	いいえ
香港	HK	852	[いいえ]	はい	はい	はい
ハンガリー	HU	36	[いいえ]	はい	いいえ	はい
I						
アイスランド	IS	354	いいえ	いいえ	はい	いいえ
インド	IN	91	はい	はい [4]	要登録 [3]	はい
インドネシア	ID	62	いいえ	いいえ	はい	いいえ
イラク	IQ	964	いいえ	いいえ	はい	いいえ

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
アイルランド	IE	353	[いいえ]	はい	はい	はい
マン島	IM	44-1624	いいえ	いいえ	はい	いいえ
イスラエル	IL	972	[いいえ]	はい	はい	はい
イタリア	IT	39	はい	はい	はい	はい
コートジボワール	CI	225	いいえ	いいえ	はい	いいえ
J						
ジャマイカ	JM	1-876	いいえ	いいえ	はい	いいえ
日本	JP	81	はい	はい	はい	はい
ジャージー	JE	44-1534	[いいえ]	はい	はい	はい
ヨルダン	JO	962	いいえ	[いいえ]	要登録 [1]	[いいえ]
K						
カザフスタン	KZ	7	いいえ	いいえ	はい	いいえ
ケニア	KE	254	いいえ	いいえ	はい	いいえ
コソボ共和国	XK	383	いいえ	いいえ	はい	いいえ
クウェート	KW	965	いいえ	[いいえ]	要登録 [1]	[いいえ]
キルギスタン	KG	996	いいえ	いいえ	はい	いいえ

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
L						
ラオス	LA	856	いいえ	いいえ	はい	いいえ
ラトビア	LV	371	いいえ	いいえ	はい	いいえ
レバノン	LB	961	いいえ	いいえ	はい	いいえ
レソト	LS	266	いいえ	いいえ	はい	いいえ
リベリア	LR	231	[いいえ]	はい	いいえ	
リビア	LY	218	いいえ	いいえ	はい	いいえ
リヒテンシュタイン	LI	423	いいえ	いいえ	はい	いいえ
リトアニア	LT	370	[いいえ]	はい	はい	はい
ルクセンブルグ	LU	352	[いいえ]	はい	はい	はい
M						
マカオ	MO	853	いいえ	いいえ	はい	いいえ
マケドニア	MK	389	いいえ	いいえ	はい	いいえ
マダガスカル	MG	261	いいえ	いいえ	はい	いいえ
マラウイ	MW	265	いいえ	いいえ	はい	いいえ
マレーシア	MY	60	はい	いいえ	いいえ	はい
モルジブ	MV	960	いいえ	いいえ	はい	いいえ

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
マリ	ML	223	いいえ	いいえ	はい	いいえ
マルタ	MT	356	いいえ	いいえ	はい	いいえ
マーシャル諸島共和国	MH	692	いいえ	いいえ	いいえ	[いいえ]
マルチニーク	MQ	596	いいえ	いいえ	はい	いいえ
モーリタニア	MR	222	いいえ	いいえ	はい	いいえ
モーリシャス	MU	230	いいえ	いいえ	はい	いいえ
マヨット	YT	262	いいえ	いいえ	はい	いいえ
メキシコ	MX	52	はい	いいえ	いいえ	はい
ミクロネシア連邦	FM	691	いいえ	いいえ	いいえ	[いいえ]
モルドバ	MD	373	いいえ	いいえ	はい	いいえ
モナコ	MC	377	いいえ	いいえ	いいえ	[いいえ]
モンゴル	MN	976	いいえ	いいえ	はい	いいえ
モンテネグロ	ME	382	いいえ	いいえ	はい	いいえ
モントセラト	MS	1-664	いいえ	いいえ	はい	いいえ
モロッコ	MA	212	はい	いいえ	はい	はい

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
モザンビーク	MZ	258	いいえ	いいえ	いいえ	[いいえ]
ミャンマー	MM	95	[いいえ]	はい	はい	はい
N						
ナミビア	NA	264	いいえ	いいえ	はい	いいえ
ネパール	NP	977	いいえ	いいえ	はい	いいえ
オランダ	NL	31	はい	はい	はい	はい
オランダ領アンティル	AN	599	いいえ	いいえ	はい	いいえ
ニューカレドニア	NC	687	いいえ	いいえ	はい	いいえ
ニュージーランド[6]	NZ	64	はい	いいえ	いいえ	はい
ニカラグア	NI	505	いいえ	いいえ	いいえ	[いいえ]
ニジェール	NE	227	いいえ	いいえ	はい	いいえ
ナイジェリア	NG	234	いいえ	いいえ	はい	いいえ
ニウエ	NU	683	いいえ	いいえ	はい	いいえ
ノルウェー	NO	47	[いいえ]	はい	はい	はい
O						
オマーン	OM	968	いいえ	いいえ	いいえ	該当なし

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
P						
パキスタン	PK	92	いいえ	いいえ	はい	該当なし
パレスチナ	PS	970	いいえ	いいえ	はい	いいえ
パナマ	PA	507	いいえ	いいえ	はい	いいえ
パプアニューギニア	PG	675	いいえ	いいえ	はい	いいえ
パラグアイ	PY	595	いいえ	いいえ	いいえ	[いいえ]
ペルー	PE	51	はい	いいえ	いいえ	はい
フィリピン	PH	63	[いいえ]	はい	要登録 [1]	はい
ポーランド	PL	48	[いいえ]	はい	はい	はい
ポルトガル	PT	351	[いいえ]	はい	はい	はい
プエルトリコ	PR	1-797、1-939	いいえ	いいえ	いいえ	はい
Q						
カタール	QA	974	いいえ	[いいえ]	要登録 [1]	[いいえ]
R						
コンゴ共和国	CG	242	いいえ	いいえ	いいえ	[いいえ]
レユニオン (フランス)	RE	262	いいえ	いいえ	はい	いいえ

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
ルーマニア	RO	40	[いいえ]	はい	はい	はい
ロシア	RU	7	はい	いいえ	要登録 [1]	はい
ルワンダ	RW	250	いいえ	いいえ	はい	いいえ
S						
セントクリストファーネイビス	KN	1-869	いいえ	いいえ	いいえ	[いいえ]
セントルシア	LC	1-758	いいえ	いいえ	いいえ	[いいえ]
サモア	WS	685	いいえ	いいえ	はい	いいえ
サンマリノ	SM	378	いいえ	いいえ	はい	いいえ
サントメ・プリンシペ民主共和国	ST	239	いいえ	いいえ	はい	いいえ
サウジアラビア	SA	966	[いいえ]	はい [4]	要登録 [1]	[いいえ]
セネガル	SN	221	いいえ	いいえ	はい	いいえ
セルビア	RS	381	いいえ	いいえ	はい	いいえ
セイシェル	SC	248	いいえ	いいえ	はい	いいえ
シエラレオネ	SL	232	いいえ	いいえ	はい	いいえ

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
シンガポール	SG	65	はい	はい	はい[5]	はい
スロバキア	SK	421	[いいえ]	はい	はい	いいえ
スロベニア	SI	386	いいえ	いいえ	はい	いいえ
ソロモン諸島	SB	677	いいえ	いいえ	はい	いいえ
ソマリア	SO	252	いいえ	いいえ	はい	いいえ
南アフリカ	ZA	27	はい	はい	いいえ	はい
韓国	KR	82	いいえ	いいえ	いいえ	[いいえ]
南スーダン	SS	211	いいえ	いいえ	はい	いいえ
スペイン	ES	34	はい	はい	はい	はい
スリランカ	LK	94	いいえ	[いいえ]	要登録 [1]	[いいえ]
スリナム	SR	597	いいえ	いいえ	はい	いいえ
スワジランド	SZ	268	いいえ	いいえ	はい	いいえ
スウェーデン	SE	46	はい	はい	はい	はい
スイス	CH	41	[いいえ]	はい	はい	はい
T						
台湾	TW	886	[いいえ]	はい	いいえ	はい

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
タジキスタン	TJ	992	いいえ	いいえ	はい	いいえ
タンザニア	TX	255	いいえ	いいえ	はい	いいえ
タイ	TH	66	[いいえ]	はい	要登録 [1]	はい
東ティモール	TL	670	いいえ	いいえ	はい	いいえ
トーゴ	TG	228	いいえ	いいえ	はい	いいえ
トンガ	TO	676	いいえ	いいえ	はい	いいえ
トリニダード・トバゴ	TT	1-868	いいえ	いいえ	はい	いいえ
チュニジア	TN	216	いいえ	いいえ	はい	いいえ
トルコ	TR	90	いいえ	[いいえ]	要登録 [1]	[いいえ]
トルクメニスタン	TM	993	いいえ	いいえ	いいえ	[いいえ]
タークスおよびカイコス諸島	TC	1-649	いいえ	いいえ	はい	いいえ
ツバル	TC	688	いいえ	いいえ	はい	いいえ
U						
ウガンダ	UG	256	いいえ	いいえ	はい	いいえ
ウクライナ	UA	380	[いいえ]	はい	はい	はい

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
アラブ首長国連邦 (UAE)	AE	971	はい	はい	要登録 [1]	はい
英国	GB	44	はい	はい	はい	はい
アメリカ	US	1	はい	はい	いいえ	はい
ウルグアイ	UY	598	はい	いいえ	いいえ	はい
ウズベキスタン	UZ	998	いいえ	いいえ	はい	いいえ
V						
バヌアツ	VU	678	いいえ	いいえ	はい	いいえ
ベネズエラ	VE	58	いいえ	いいえ	いいえ	[いいえ]
ベトナム	VN	84	いいえ	[いいえ]	要登録 [1]	[いいえ]
英領バージン諸島	VG	1-284	いいえ	いいえ	はい	いいえ
米領バージン諸島	VI	1-340	いいえ	いいえ	いいえ	はい
W						
X						
Y						
イエメン	YE	967	いいえ	いいえ	はい	いいえ
Z						

国または地域	ISO コード	ダイヤルコード	ショートコードをサポートする	ロングコードをサポートする	送信者 ID をサポートする	双方向 SMS をサポートする
ザンビア	ZM	260	いいえ	いいえ	はい	いいえ
ジンバブエ	ZW	263	いいえ	いいえ	はい	いいえ

メモ

1. 送信者は、事前に登録されたアルファベットの送信者 ID を使用する必要があります。から送信者 ID をリクエストするには AWS Support、を参照してください。Amazon SNS で SMS メッセージングの送信者 ID をリクエストする一部の国では、承認を得るために、送信者は特定の要件を満たすか、特定の制限に従う必要があります。このような場合は、Sender ID のリクエストを送信した後に、 AWS Support 追加情報についてお客様に連絡することがあります。

2. 送信者は、送信する予定のメッセージのタイプごとに、事前に登録されたテンプレートを使用する必要があります。送信者がこの要件を満たしていない場合、送信者のメッセージはブロックされます。テンプレートを登録するには、を使用して Amazon SNS SMS ケースを開きます。 AWS Supportケースを作成する際に、送信者 ID を要求するときと同じ情報を提供します。詳細については、「Amazon SNS で SMS メッセージングの送信者 ID をリクエストする」を参照してください。一部の国では、承認を得るために、特定の要件を満たすか、特定の制限に従う必要があります。このような場合、 AWS Support 追加情報の入力を求められることがあります。

   > **Note**
   >
   > 中国にメッセージを送信するには、 AWS Support まずテンプレートを登録して承認を受ける必要があります。

3. 送信者は、事前に登録されたアルファベットの送信者 ID を使用する必要があります。追加の登録手順を経る必要があります。詳細については、「インドの送信者 ID 登録要件」を参照してください。

4. これらの国のロングコードは、インバウンドメッセージングにのみ対応しています。つまり、これらのロングコードは、受信者へのメッセージ送信には使用できませんが、受信者からのメッセージは受信できます。送信者 ID はメッセージの送信にのみ対応しているため、アルファベット

の送信者 ID を使用してメッセージを送信する場合、ロングコードは受信者がオプトアウトするのに便利な方法です。

5. Amazon SNS は、シンガポール SMS 送信者 ID レジストリ (SSIR) に登録された送信者 ID を使用して、シンガポールに SMS トラフィックを送信できます。SSIR は、シンガポールの情報通信メディア開発局 (IMDA) が作成したレジストリです。シンガポールの送信者 ID を使用するための要件の詳細については、「シンガポールの送信者 ID 登録要件」を参照してください。

   未登録の送信者 ID、またはショートコードやロングコードなどの代替の送信元 ID タイプを使用して、シンガポールで SMS トラフィックを送信することもできます。

6. 専用ショートコードがない場合でも、Amazon SNS はショートコードの共有プールを使用してニュージーランドの受信者にメッセージを送信しようとします。共有番号に関する現地の通信事業者の制限により、このような共有番号での配信性能はベストエフォートベースで決定されます。このため、Amazon SNS では、ニュージーランドに送信されるすべてのトラフィックに専用のショートコードを用意することを強くお勧めします。URL を含むメッセージは、専用のショートコードプロセスを通じて許可リストに登録する必要があります。ショートコードの購入については、「Amazon SNS による SMS メッセージングの専用ショートコードをリクエストする」を参照してください。

SMS ベストプラクティス

携帯電話ユーザーは、未承諾 SMS メッセージに対する許容度が極めて低い傾向があります。未承諾 SMS キャンペーンの反応率はほぼ常に低くなるため、投資効果は低くなります。

さらに、携帯電話キャリアは大量 SMS の送信者を継続的に監査しています。未承諾メッセージを送信していると判断した番号からのメッセージは、調整されたりブロックされたりします。

未承諾コンテンツの送信は、AWS の適正利用規約にも違反します。Amazon SNS チームは、SMS キャンペーンを定期的に監査しており、未承諾メッセージを送信していると思われる場合はメッセージの送信能力を調整したりブロックしたりすることがあります。

最後に、多くの国、リージョン、管轄区域では、未承諾 SMS メッセージの送信に多額の罰金を科しています。例えば、米国の Telephone Consumer Protection Act (TCPA) には、消費者が未承諾メッセージを 1 通受け取るたびに 500 ~ 1,500 USD の損害賠償を受け取ることができる (送信者が支払う) と規定されています。

このセクションでは、顧客エンゲージメントを高め、罰金を回避するために役立つ可能性があるいくつかのベストプラクティスについて説明します。ただし、このセクションには法的なアドバイスは含

まれていないことに注意してください。法的なアドバイスを受けるには、弁護士に相談してください。

トピック

法律、規制、および通信事業者の要件の遵守

あなたの顧客が所在する地域の法および規制に違反すると、高額な罰金と罰が課されます。このためにも、事業を展開する各国や各地域の SMS メッセージングに関する法律を理解することは非常に重要です。

次のリストには、世界中の主要なマーケットにおける SMS コミュニケーションに適用される重要な法律のリンクが含まれています。

- 米国 : 1991 年の電話利用者保護法 (TCPA) は、特定の種類の SMS メッセージに適用されます。詳細については、規則および規制を連邦通信委員会のウェブサイトで参照してください。
- 英国 : 2003 年のプライバシーおよび電子通信に関する規制(EC 指令) (PECR) は、 特定の種類の SMS メッセージに適用されます。詳細については、英国情報コミッショナーオフィスのウェブサイトで「What are PECR?」を参照してください。
- EU: 2002 年のプライバシーおよび電子通信に関する規則指令 (ePrivacy 指令と呼ばれることもある) は、特定の種類の SMS メッセージに適用されます。詳細については、法律全文 を Europa.eu ウェブサイトで参照してください。
- カナダ: インターネットおよび無線スパム対策法 (カナダのアンチスパム法あるいは CASL) は、特定の種類の SMS メッセージに適用されます。詳細については、法律全文 をカナダ議会のウェブサイトで参照してください。
- 日本: 特定電子メールの送信の適正化等に関する法律は、特定の種類の SMS メッセージに適用される場合があります。詳細については、日本の迷惑メール対策を総務省のウェブサイトで参照してください。

送信者として、会社または組織がこれらの国のいずれかに拠点を置いていない場合でも、これらの法律が適用される場合があります。このリストの一部の法規制は迷惑 E メールあるいは電話を対象に元々制定されたものもありますが、SMS メッセージにも同様に適用するとの見解や拡張解釈が行われています。その他の国々や地域では、SMS メッセージの送信に関する独自の法規制を設けています。顧客が所在する各国や各地域の弁護士に相談して、法的なアドバイスを受けてください。

多くの国では、現地の通信事業者がネットワーク上を流れるトラフィックの種類を決定する最終権限を保有しています。つまり、通信事業者は、現地の法律の最低要件より厳しい制限を SMS コンテンツに課す可能性があります。

許可を取得する

送信する予定のメッセージについて特定のタイプの受信を明示的に要求していない受信者には、決してメッセージを送信しないでください。同じ会社内の組織間であっても、オプトインリストを共有しないでください。

受信者がオンラインフォームを使用してメッセージの受信にサインアップできる場合、本人が気づかずに自動スクリプトがサブスクライブすることを防止するシステムを追加してください。また、1 回のセッションでユーザーが電話番号を送信できる回数を制限する必要があります。

SMS オプトインリクエストを受け取ったら、メッセージの受信を希望することを確認するメッセージを受信者に送信します。受信者がサブスクリプションを確認するまで、追加のメッセージを受信者に送信しないでください。サブスクリプション確認メッセージの例は次のとおりです。

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may
apply. Reply HELP for help, STOP to cancel.
```

日付、時刻、各オプトインリクエストおよび確認のソースが含まれる記録を保持してください。これは、通信事業者または規制機関から依頼された場合に役立つことがあり、顧客リストの定期的な監査の実施にも役立つことがあります。

オプトインワークフロー

場合によっては (米国の通話料無料やショートコード登録など)、携帯電話会社からオプトインワークフロー全体のモックアップまたはスクリーンショットの提供が求められます。モックアップまたはスクリーンショットは、受信者が完了するオプトインワークフローによく似ている必要があります。

モックアップまたはスクリーンショットには、最高レベルのコンプライアンスを維持するために必要な以下の開示事項をすべて含める必要があります。

必要な開示

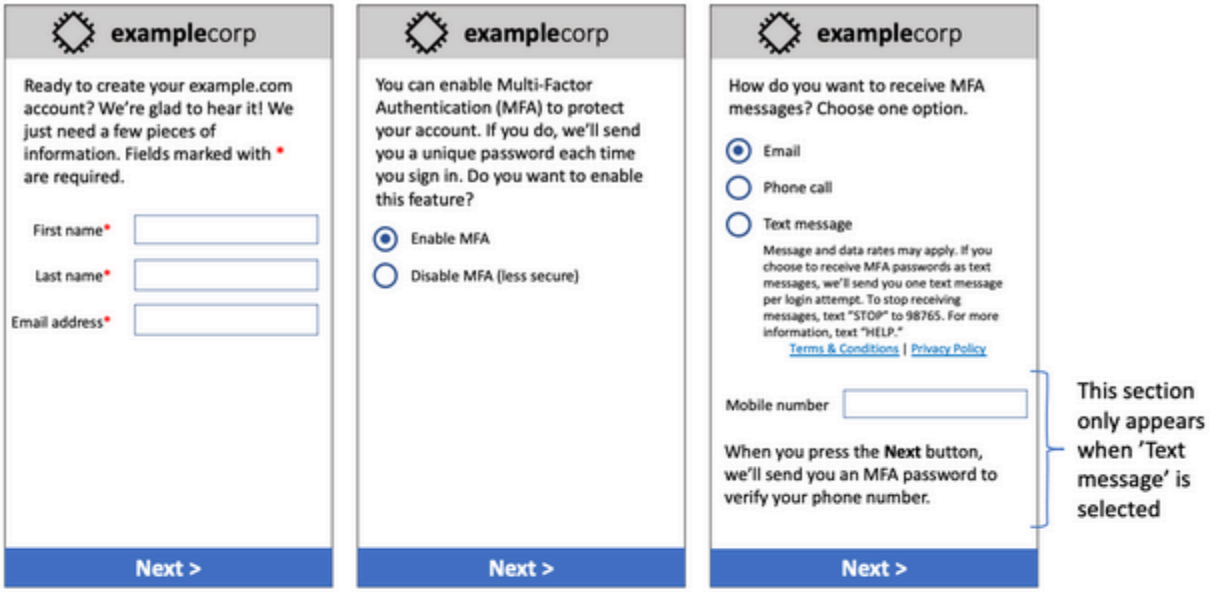
- プログラムを通じて送信するメッセージングユースケースの説明。
- 「メッセージ料金とデータ料金が適用される場合があります」というフレーズ。
- 受信者がメッセージを受け取る頻度の提示。例えば、定期的なメッセージングプログラムでは、「週に 1 通のメッセージ」と表示される場合があります。ワンタイムパスワードまたは多要素認証のユースケースでは、「メッセージの頻度は異なる」または「ログイン試行ごとに 1 通のメッセージ」と表示される場合があります。
- 利用規約とプライバシーポリシー文書へのリンク。

コンプライアンス違反オプトインの一般的な拒否理由

- 提供された会社名がモックアップまたはスクリーンショットに記載されているものと一致しない場合。明確ではない関係については、オプトインワークフローの説明で説明する必要があります。
- メッセージが受信者に送信されるように見えても、送信前に明示的に同意が得られない場合。すべてのメッセージには、明示的な同意が必要です。
- サービスにサインアップするためにテキストメッセージの受信が必要と思われる場合。ワークフローが、電子メールや音声通話などの別の形式でオプトインメッセージを受信する代替手段を提供していない場合、これは準拠しません。
- オプトイン言語がすべて利用規約に記載されている場合。開示内容は、リンク先のポリシー文書の中に入れるのではなく、オプトイン時に必ず受信者に提示する必要があります。

- 顧客があるタイプのメッセージを受信することに同意し、他のタイプのテキストメッセージを送信する場合。例えば、顧客はワンタイムパスワードの受信には同意しましたが、投票メッセージやアンケートメッセージも送信されます。
- 必要な開示 (上記) が受信者に提示されない場合。

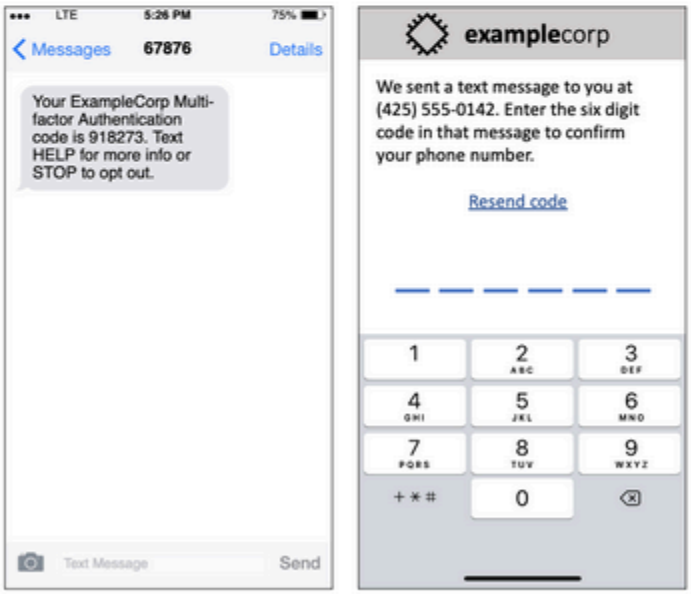
次の例は、多要素認証のユースケースに関する携帯端末通信業者の要件に準拠しています。



1. User provides basic account information.

2. User decides whether to enable MFA.

3. If MFA enabled, user chooses how to receive MFA token.



4. If user chooses to receive MFA token by text, send a token.

5. User enters MFA token to verify phone number.

多要素認証のユースケースのモックアップ

完成したテキストと画像が含まれ、オプトインフロー全体が注釈付きで表示されます。オプトインフローでは、顧客はテキストメッセージの受信に同意するために、明確な意図的行動をとり、必要な開示事項をすべて含める必要があります。

その他のオプトインワークフロータイプ

携帯端末通信業者は、上記の規定に準拠している場合、口頭または書面によるオプトインなど、アプリケーションやウェブサイト以外のオプトインワークフローも受け入れます。コンプライアンスに準拠したオプトインワークフローと、口頭または書面によるスクリプトは、特定のメッセージタイプを受信することについて、受信者から明示的な同意を得ます。例としては、サービスデータベースに記録する前にサポートエージェントが同意を得るために使用する口頭スクリプトや、販促用チラシに記載されている電話番号などがあります。これらのオプトインワークフロータイプのモックアップには、オプトインスクリプト、マーケティング資料、または数字が収集されているデータベースのスクリーンショットを提供できます。オプトインが明確ではない場合やユースケースが一定量を超える場合、携帯端末通信業者はこれらのユースケースに関して追加の質問をすることがあります。

古いリストには送信しないでください

人は電話番号を頻繁に変更します。2 年前に連絡先として同意を得た電話番号は、今では他の誰かが使用しているかもしれません。新しいメッセージングプログラムに古い電話番号リストを使用しないでください。使用した場合、電話番号がすでに使用されていないため、メッセージが送信されない場合や、そもそも同意したことを忘れてオプトアウトする人がいる可能性があります。

顧客リストを監査する

定期的な SMS キャンペーンを送信している場合、顧客リストを定期的に監査してください。顧客リストを監査すると、メッセージを受信したい顧客だけがメッセージを確実に受信するようになります。

リストを監査するときは、オプトインしている各顧客に、サブスクライブしていることを再確認するメッセージを送信し、サブスクライブ解除に関する情報を提供します。再確認メッセージの例は次のとおりです。

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

レコードを保持する

SMS メッセージを送信するよう各顧客よりリクエストされた日時と、各顧客に送信したメッセージを示すレコードを保持します。世界の多くの国や地域では、SMS の送信者がこれらのレコードを簡単に取り出せるように維持する必要があります。また、この情報は、携帯端末通信事業者より随時リクエストされる場合があります。提供する情報は、国または地域によって異なります。レコード維持の要件の詳細は、顧客が所在する国または地域ごとの商用 SMS メッセージングの規制を確認します。

場合によっては、キャリアや規制機関より、顧客がメッセージの受け取りを選択した事実の提供を求められることがあります。このような場合は、キャリアや機関が必要な情報のリストについて、AWS Support より連絡があります。必要な情報を提供しない場合は、以降の SMS メッセージを送信する機能が一時停止される場合があります。

メッセージは明確、正直、簡潔にしてください

SMS はユニークなメディアです。メッセージあたり 160 文字という制限があるため、メッセージは簡潔にする必要があります。電子メールなど、他の通信チャネルで使用する手法が SMS チャネルでは適用できない場合があり、SMS メッセージで使用すると不正または詐欺的に思われる場合もあります。メッセージの内容がベストプラクティスと一致しない場合、受信者はメッセージを無視する可能性があります。最悪の場合、携帯電話通信業者がメッセージをスパムと見なし、その電話番号からのメッセージを今後ブロックする可能性があります。

このセクションでは、効果的な SMS メッセージ本文を作成するためのヒントやアイデアをいくつか紹介します。

自分が送信者であることを明記してください

受信者は、メッセージがあなたから送信されたものであることをすぐに理解できる必要があります。このベストプラクティスに従って、送信者は各メッセージの最初に識別名 (「プログラム名」) を含めます。

この操作はしないでください。

```
Your account has been accessed from a new device. Reply Y to confirm.
```

代わりにこれを試してください:

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

メッセージを個人から個人へのメッセージのように記載しない

マーケティング担当者の中には、メッセージを個人から送信されたように見せて、SMS メッセージに個人的なタッチを加えたいと思う人もいます。ただし、この手法を使用すると、メッセージがフィッシング詐欺のように受け取られる場合があります。

この操作はしないでください。

```
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.
```

代わりにこれを試してください:

```
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.
```

お金の話には注意する

詐欺師は、お金を貯めたり、受け取りたいという人の欲求につけ込むことがよくあります。「話がうますぎる」と思わせるようなオファーを記載してはいけません。人を欺くためにお金の誘惑を使用してはいけません。通貨記号を使ってお金を示してはいけません。

この操作はしないでください。

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

代わりにこれを試してください:

```
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.
```

必要な文字のみを使用する

ブランドは、多くの場合において、メッセージに ™ や® などの商標記号を含めることで商標を保護する傾向があります。しかし、これらの記号は、160 文字の SMS メッセージに含めることができる (GSM アルファベットと呼ばれる) 標準文字セットには含まれていません。これらの文字のいずれかを含むメッセージを送信すると、メッセージは別の文字エンコーディングシステムを使用して自動送信されます。このシステムでは、メッセージパートごとに 70 文字までしかサポートされません。その結果、メッセージが複数に分割される可能性があります。送信するメッセージパートごとに料金が請求されるため、メッセージ全体の送信にかかる費用が予想よりも高くなる可能性があります。さらに、受信者は 1 つのメッセージではなく、複数の連続したメッセージを受信する場合があります。SMS の文字エンコーディングの詳細については、「Amazon SNS の SMS の文字数制限について」を参照してください。

この操作はしないでください。

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

代わりにこれを試してください:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> 上記の 2 つの例はほとんど同じですが、最初の例には GSM アルファベットではない登録商標記号 (®) が含まれています。その結果、最初の例は 2 つのメッセージパートとして送信され、2 番目の例は 1 つのメッセージパートとして送信されます。

有効で安全なリンクを使用する

メッセージにリンクが含まれている場合は、そのリンクが機能することを再確認してください。企業ネットワーク外のデバイスでリンクをテストして、リンクが正しく解決されることを確認します。SMS メッセージは 160 文字に制限されているため、URL が非常に長いと複数のメッセージに分割される可能性があります。短縮 URL を指定するには、リダイレクトドメインを使用する必要があります。ただし、tinyurl.com や bitly.com などの無料のリンク短縮サービスを使用しないでください。通信事業者は、そのドメイン上のリンクを含むメッセージをフィルタリングする傾向があります。リンクが会社または組織の排他的使用専用のドメインを指している限り、有料のリンク短縮サービスを使用できます。

この操作はしないでください。

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

代わりにこれを試してください:

```
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.
```

使用する略語数を制限する

SMS チャネルには 160 文字の制限があるため、一部の送信者は、メッセージに略語を多用する必要があると考えます。しかし、略語を使いすぎると、多くの読者は企業からのものではないと受け止められ、一部のユーザーがメッセージをスパムとして報告する可能性があります。略語を多用せずに、首尾一貫したメッセージを書くことは至って可能です。

この操作はしないでください。

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

代わりにこれを試してください:

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

適切に応答する

受信者がメッセージに返信したときは、有用な情報で応答するようにしてください。例えば、顧客がメッセージの 1 つにキーワード「HELP」で反応した場合、サブスクライブしているプログラム、毎月送信されるメッセージの数、詳細情報が必要な場合の連絡方法に関する情報を送信します。HELP レスポンスメッセージの例は次のとおりです。

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

顧客がキーワード「STOP」で返信した場合、今後はメッセージが送信されないことを知らせます。STOP レスポンスメッセージの例は次のとおりです。

```
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.
```

エンゲージメントに基づく送信を調整する

顧客の優先順位は時間の経過とともに変わる可能性があります。顧客がメッセージを必要としなくなった場合、メッセージを完全にオプトアウトしたり、メッセージを未承諾として報告したり可能性すらあります。このため、顧客とのエンゲージメントに基づいて送信手続きを調整することは重要です。

メッセージにめったに反応しない顧客の場合、メッセージの頻度を調整する必要があります。例えば、反応の多い顧客にメッセージを毎週送信している場合、反応の少ない顧客用に別の毎月のダイジェストを作成できます。

最後に、まったく反応のない顧客を顧客リストから削除します。このステップにより、顧客がメッセージを不満に感じることがなくなります。また、コストを削減し、送信者としての評判を維持することもできます。

適切な時間に送信する

通常の営業時間中にのみメッセージを送信します。メッセージを夕食や深夜の時間帯に送信している場合、顧客が邪魔をされないようにリストからサブスクリプション解除する可能性が高くなります。さらに、顧客がすぐに反応できないときに SMS メッセージを送信しても意味がありません。

キャンペーンやジャーニーを多くの受信者に送信する場合は、発信元番号のスループットレートを再確認してください。受信者数をスループットレートで割ると、すべての受信者にメッセージを送信するのにかかる時間が分かります。

複数チャンネルでの重複を回避する

キャンペーンで、複数の通信チャンネル (E メール、SMS、プッシュメッセージ) を使用している場合、各チャンネルで同じメッセージを送信しないでください。同じメッセージを複数のチャンネルで同時に送信すると、顧客は送信動作を有用ではなく迷惑だと受け取る可能性が高くなります。

専用ショートコードを使用する

ショートコードを使用している場合、ブランドおよびメッセージタイプごとに別個のショートコードを維持します。例えば、会社に 2 つのブランドがある場合、それぞれ別のショートコードを使用します。同様に、トランザクションメッセージとプロモーションメッセージの両方を送信している場合、メッセージタイプごとに別のショートコードを使用します。ショートコードをリクエストする方法の詳細については、「Amazon SNS による SMS メッセージングの専用ショートコードをリクエストする」を参照してください。

送信先の電話番号を確認する

Amazon Pinpoint を通じて SMS メッセージを送信すると、送信したメッセージパートごとに課金されます。メッセージパートごとに支払う料金は、受信者の国や地域によって異なります。SMS の料金の詳細については、「Amazon SNS の料金」を参照してください。

Amazon SMS が SMS メッセージを送信するリクエストを受け入れた場合 (SendMessages API、またはキャンペーンまたはジャーニーが開始された結果)、そのメッセージの送信に対して課金されます。このステートメントは、たとえ意図した受信者が実際にメッセージを受け取らなかったとしても有効です。例えば、相手の電話番号が使われていなかったり、送信先の電話番号が有効な携帯電話番号でなかったとしても、メッセージの送信は課金対象となります。

Amazon SNS は、SMS メッセージの有効な送信リクエストを受け付け、配信を試みます。このため、メッセージを送信する電話番号が有効な携帯電話番号であることを検証する必要があります。Amazon SNS 電話番号検証サービスを利用することで、電話番号が有効かどうか、どのような

種類の電話番号か (携帯電話、固定電話、VoIPなど) を検証することができます。詳細については、『Amazon Pinpoint デベロッパーガイド』の「Validating phone numbers in Amazon Pinpoint」を参照してください。

冗長性を考慮した設計

ミッションクリティカルなメッセージングプログラムでは、Amazon SNS を複数の AWS リージョン で設定することをお勧めします。Amazon SNS は、複数の AWS リージョン で利用できます。Amazon SNS を使用できるリージョンの一覧については、「AWS 全般のリファレンス」を参照してください。

ショートコード、ロングコード、通話料無料、10DLC 番号など、SMS メッセージに使用する電話番号は AWS リージョン で複製することはできません。そのため、Amazon SNS を複数のリージョンで使用するには、Amazon SNS を使用するリージョンごとに個別の電話番号をリクエストする必要があります。例えば、米国の受信者にショートコードを使用する場合、使用する予定の各 AWS リージョン で、ショートコードごとに個別のショートコードをリクエストする必要があります。

一部の国では、冗長性を高めるために複数のタイプの電話番号を使用することもできます。例えば、米国では、ショートコード、10DLC 番号、通話料無料の番号をリクエストすることができます。これらの電話番号タイプはそれぞれ、受信者へのルートが異なります。電話番号の種類が複数あることは (同じ AWS リージョン または複数の AWS リージョン にまたがる場合でも)、追加の冗長層を提供し、回復力の向上に役立ちます。

SMS の制限と規制

SMS の制限と規制については、Amazon Pinpoint ユーザーガイドの「Amazon Pinpoint での SMS の制限と規制」を参照してください。

オプトアウトキーワードの管理

SMS 受信者は、各自のデバイスを使用して、キーワードで返信することでメッセージをオプトアウトできます。詳細については、「SMS メッセージの受信をオプトアウトする」を参照してください。

CreatePool

`CreatePool` API アクションを使用して新しいプールを作成し、そのプールと指定した送信元アイデンティティとを関連付けます。詳細については、Amazon Pinpoint SMS および音声 API の「CreatePool」を参照してください。

PutKeyword

`PutKeyword` API アクションを使用して、送信元の電話番号またはプールのキーワード設定を作成または更新します。詳細については、Amazon Pinpoint SMS および音声 API の「PutKeyword」を参照してください。

番号設定の管理

[SMS and voice settings] (SMS と音声設定) ページの [Number settings] (番号設定) セクションのオプションを使用し、AWS サポートにリクエストしてアカウントに割り当てた専用のショートコードとロングコードの設定を管理できます。詳細については、Amazon Pinpoint ユーザーガイドの「番号設定の管理」を参照してください。

Amazon SNS の SMS の文字数制限について

1 回の SMS メッセージには、最大 140 バイトの情報を含めることができます。1 回の SMS メッセージに使用できる文字数は、メッセージに含む文字の種類によって異なります。

GSM 7-bit アルファベットとしても知られる「GSM 03.38 文字セット」のみメッセージに使用されている場合は、最大 160 文字含めることができます。メッセージに GSM 03.38 文字セット以外の文字を使用する場合は、最大 70 文字を含めることができます。SMS メッセージを送信する場合は、Amazon SNS によって最も効率的なエンコードが自動的に判断されます。

メッセージに最大文字数を超える文字を含めると、メッセージは複数のパートに分割されます。メッセージが複数のパートに分割されると、各パートには、その前のメッセージパートに関する追加情報が含まれます。受取人のデバイスは、このように分割されたメッセージパートを受信すると、この追加情報を使用して、すべてのメッセージパートを正しい順で表示します。受取人の携帯通信事業者やデバイスによっては、複数のメッセージが、1 つのメッセージとして表示されたり、個別のメッセージのシーケンスとして表示されたりする場合があります。その結果として、各メッセージパートの文字数は 153 (GSM 03.38 文字のみを含むメッセージの場合) または 67 (他の文字を含むメッセージの場合) に減少します。SMS の長さ計算ツールを使用すると、メッセージを送信する前にメッセージに含まれるメッセージパートの数を推定できます。これらのツールのいくつかはオンラインで入手できます。メッセージの最大サイズは、GSM 1600 文字または非 GSM 630 文字です。スループットとメッセージサイズの詳細については、Amazon Pinpoint ユーザーガイドの「SMS character limits in Amazon Pinpoint」(Amazon Pinpoint の SMS の文字数制限について) を参照してください。

送信する各メッセージのメッセージパート数を確認するには、まずイベントストリーム設定を有効にする必要があります。有効にすると、メッセージが受取人の携帯電話会社に配信されたときに、Amazon SNS によって `_SMS.SUCCESS` イベントが生成されます。`_SMS.SUCCESS` イベントレ

コードには、`attributes.number_of_message_parts` という属性が含まれています。この属性は、メッセージ内のメッセージパート数を指定します。

> **Important**
>
> 複数のメッセージパートを含むメッセージを送信すると、メッセージ内のメッセージパート数に応じて課金されます。

GSM 03.38 文字セット

GSM 03.38 文字セットに含まれるすべての文字を次のテーブルに示します。次のテーブルに示す文字のみを含むメッセージを送信する場合、そのメッセージには最大 160 文字までを含めることができます。

GSM 03.38 標準文字												
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
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						

GSM 03.38 文字セットには、前述のテーブルに示す記号に加えて、複数の記号が含まれています。ただし、これらの文字には、表示されない エスケープ文字も含まれているため、それぞれ 2 文字としてカウントされます。

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

最後に、GSM 03.38 文字セットには、次の非表示の文字も含まれています。

- 空白文字。
- 改行制御。テキストのある行の行末であるとともに、次の行の先頭であることを示します。
- キャリッジリターン制御。テキストの行の先頭に移動します (通常は次の改行文字)。
- エスケープ制御。前述のリストの文字に自動的に追加されます。

メッセージの例

このセクションでは、SMS メッセージの例をいくつか示します。例ごとに、メッセージのメッセージパート数と合計文字数を示します。

例 1: GSM 03.38 アルファベットの文字のみを含む長いメッセージ

次のメッセージには、GSM 03.38 アルファベットの文字のみが含まれています。

```
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.
```

上のメッセージには 180 文字が含まれているため、複数のメッセージパートに分割する必要があります。メッセージを複数のメッセージパートに分割した場合、各パートには 153 文字の GSM 03.38 を含めることができます。その結果、このメッセージは 2 つのメッセージパートとして送信されます。

例 2: マルチバイト文字を含むメッセージ

次のメッセージには、GSM 03.38 アルファベットではない、複数の中国語の文字が含まれています。

```
##########################################################·####1994#7#########
```

上のメッセージには 71 文字が含まれています。ただし、メッセージ内のほとんどすべての文字は GSM 03.38 アルファベットではないため、2 つのメッセージパートとして送信されます。メッセージパートごとに最大 67 文字を含めることができます。

例 3: GSM 以外の文字を 1 つ含むメッセージ

次のメッセージには、GSM 03.38 アルファベットに属さない文字が 1 つ含まれています。この例で、その文字は閉じる単一引用符 (') であり、通常のアポストロフィ (') とは別の文字です。Microsoft Word などのワープロアプリケーションは、アポストロフィを、閉じる単一引用符に自動的に置き換える場合があります。SMS メッセージの下書きを Microsoft Word で作成して Amazon SNS に貼り付ける場合は、これらの特殊文字を削除して、アポストロフィに置き換えてください。

```
John: Your appointment with Dr. Salazar’s office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

上のメッセージには 130 文字が含まれています。ただし、GSM 03.38 アルファベットに属さない閉じる単一引用符文字が含まれているため、メッセージは 2 つのメッセージパートとして送信されます。

このメッセージの閉じる単一引用符をアポストロフィ (GSM 03.38 アルファベットの一部) に置き換えると、メッセージは単一のメッセージパートとして送信されます。

モバイルプッシュ通知

Amazon SNS によって、モバイルデバイスのアプリケーションにプッシュ通知メッセージを直接送信できます。モバイルエンドポイントに送信されたプッシュ通知メッセージは、メッセージアラート、バッジ更新、または音声アラートとしてモバイルアプリケーションに表示できます。

トピック

- ユーザー通知の仕組み
- ユーザー通知プロセスの概要
- モバイルアプリのセットアップ
- モバイルプッシュ通知を送信する

- モバイルアプリケーションの属性
- モバイルアプリケーションのイベント
- モバイルプッシュ API アクション
- モバイルプッシュ API エラー
- モバイルプッシュ通知サービスの Amazon SNS 有効期限 (TTL) メッセージ属性を使用する
- モバイルアプリケーションでサポートされているリージョン
- モバイルプッシュ通知のベストプラクティス

ユーザー通知の仕組み

以下のサポートされているプッシュ通知サービスのいずれかを使用して、モバイルデバイスとデスクトップの両方にプッシュ通知メッセージを送信します。

- Amazon Device Messaging (ADM)
- iOS と Mac OS Xの両方向け Apple Push Notification Service (APNs)
- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging (FCM)
- Windows Phone (MPNS) 用 Microsoft Push Notification Service (MPNS)
- Windows Push Notification Services (WNS)

APN や FCM などのプッシュ通知サービスは、各アプリケーションと、サービスを使用するために登録されている関連モバイルデバイスとの接続を維持します。アプリとモバイルデバイスの登録時に、プッシュ通知サービスはデバイストークンを返します。Amazon SNS はデバイストークンを使用して、モバイルエンドポイントを作成します。ここに直接プッシュ通知メッセージを送信できます。Amazon SNS が他のプッシュ通知サービスと通信できるようにするには、プッシュ通知サービス認証情報を Amazon SNS に送信して、代理で使用できるようにします。詳細については、「ユーザー通知プロセスの概要」を参照してください。

直接プッシュ通知メッセージを送信するだけでなく、Amazon SNS を使用して、トピックにサブスクライブされているモバイルエンドポイントにメッセージを送信することもできます。このコンセプトは、「Amazon SNS とは」に説明されているように、Amazon SQS、HTTP/S、E メール、SMS などその他のエンドポイントタイプをトピックにサブスクライブするのと同じです。違いは、Amazon SNS はサブスクライブされたモバイルエンドポイントがトピックに送信されたプッシュ通知メッセージを受信するためにプッシュ通知サービスを通じて通信を行うことです。

ユーザー通知プロセスの概要

1. サポートするモバイルプラットフォームの認証情報とデバイストークンを取得します。
2. 認証情報を使用して、Amazon SNS でプラットフォームアプリケーションオブジェクト (PlatformApplicationArn) を作成します。詳細については、「プラットフォームアプリケーションを作成する」を参照してください。
3. 返された認証情報を使用して、プッシュ通知サービスから、モバイルアプリケーションおよびデバイス用のデバイストークンをリクエストします。受け取ったトークンはモバイルアプリケーションおよびデバイスの識別に使用されます。
4. デバイストークンと PlatformApplicationArn を使用し、Amazon SNS でプラットフォームエンドポイントオブジェクト (EndpointArn) を作成します。詳細については、「プラットフォームエンドポイントを作成する」を参照してください。
5. EndpointArn を使用して、モバイルデバイスのアプリケーションにメッセージを発行します。詳細については、『Amazon Simple Notification Service API リファレンス』の「モバイルデバイスへ発行する」および「公開 API」を参照してください。

モバイルアプリのセットアップ

このセクションでは、「モバイルアプリケーションの設定」Amazon SNS ユーザー通知の前提条件 で説明されている情報を使用して、を使用する方法について説明します。 AWS Management Console

トピック

- Amazon SNS ユーザー通知の前提条件
- プラットフォームアプリケーションを作成する
- プラットフォームエンドポイントを作成する
- デバイストークンまたは登録 ID を追加する
- Apple 認証方法
- Firebase Cloud Messaging (FCM) の認証方法
- Firebase クラウドメッセージング (FCM) エンドポイント管理

Amazon SNS ユーザー通知の前提条件

Amazon SNS モバイルプッシュ通知の使用を開始するには、以下が必要です。

- サポートされているプッシュ通知サービスのいずれかに接続するための認証情報セット: ADM、APN、Baidu、FCM、MPNSまたは WNS。
- モバイルアプリケーションやデバイスのデバイストークンまたは登録 ID。
- モバイルエンドポイントにプッシュ通知メッセージを送信するように設定された Amazon SNS。
- 登録済みで、サポートされているいずれかのプッシュ通知サービスを使用するように設定されたモバイルアプリケーション。

アプリケーションをプッシュ通知サービスに登録するには、いくつかのステップが必要です。Amazon SNS では、モバイルエンドポイントに直接プッシュ通知メッセージを送信するために、ユーザーがプッシュ通知サービスに対して指定する情報の一部が必要です。一般的に、プッシュ通知サービスに接続するための必須認証情報、プッシュ通知サービスから受け取ったデバイストークンまたは登録 ID(モバイルデバイスおよびモバイルアプリケーションを表すもの)、およびプッシュ通知サービスに登録されているモバイルアプリケーションが必要です。

認証情報の正確な形式はモバイルプラットフォームごとに異なりますが、どの場合も、これらの認証情報はプラットフォームに接続するときに送信する必要があります。モバイルアプリケーションごとに 1 つの認証情報セットが発行され、そのアプリケーションのどのインスタンスにメッセージを送信する際もその認証情報セットを使用する必要があります。

具体的な名前は、どのプッシュ通知サービスが使用されるかによって異なります。例えば、プッシュ通知サービスとして APN を使用する場合は、デバイストークンが必要になります。また、FCM を使用する場合は、デバイストークンに相当するものが登録 ID と呼ばれます。デバイストークンまたは登録 ID は、モバイルデバイスのオペレーティングシステムによってアプリケーションに送信される文字列です。特定のモバイルデバイスで実行されるモバイルアプリケーションのインスタンスを一意に識別するため、このアプリケーション/デバイスペアの一意識別子と考えることができます。

Amazon SNS は、認証情報 (および他のいくつかの設定) をプラットフォームアプリケーションリソースとして保存します。デバイストークン (ここでもいくつかの追加設定を使用) は、プラットフォームエンドポイントと呼ばれるオブジェクトとして表現されます。各プラットフォームエンドポイントは、1 つの特定のプラットフォームアプリケーションに属し、対応するプラットフォームアプリケーションに保存された認証情報を使用してすべてのプラットフォームエンドポイントに通信できます。

以降のセクションでは、サポートされているプッシュ通知サービスごとの前提条件が示されています。前提条件の情報を取得した後は、AWS Management Console または Amazon SNS モバイルプッシュ API を使用して、プッシュ通知メッセージを送信できます。詳細については、「ユーザー通知プロセスの概要」を参照してください。

プラットフォームアプリケーションを作成する

Amazon SNS がモバイルエンドポイントに通知メッセージを送信するためには、直接か、トピックへのサブスクリプションかにかかわらず、最初にプラットフォームアプリケーションを作成する必要があります。アプリケーションを AWS に登録したら、次のステップとして、アプリケーションとモバイルデバイス用のエンドポイントを作成します。このエンドポイントは、アプリケーションとデバイスに通知メッセージを送信するために Amazon SNS によって使用されます。

プラットフォームアプリケーションを作成するには

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションペインで、[Mobile] (モバイル) を選択して [Push notifications] (プッシュ通知) を選択します。
3. [Platform applications] (プラットフォームアプリケーション) セクションで、[Create platform application] (プラットフォームアプリケーションの作成) を選択します。

   モバイルアプリケーションを作成できる AWS リージョンのリストについては、「モバイルアプリケーションでサポートされているリージョン」を参照してください。
4. [Application name] (アプリケーション名) ボックスにアプリケーションの名前を入力します。

   アプリケーション名は大文字および小文字の ASCII 文字、数字、アンダースコア、ハイフン、およびピリオドのみで構成する必要があります。また、名前は、1～256 文字の長さである必要があります。
5. [Push notification platform] (プッシュ通知プラットフォーム) で、アプリケーションの登録先のプラットフォームを選択し、適切な認証情報を入力します。

   > **Note**
   >
   > Apple Push Notification Service (APN) プラットフォームのいずれかを使用している場合は、トークンベースまたは証明書ベースの認証のいずれかを選択し、[Choose file] (ファイルの選択) を選択して .p8 または .p12 ファイル (キーチェーンアクセスからエクスポート) を Amazon SNS にアップロードできます。
6. [プラットフォームアプリケーションの作成] を選択します。

   アプリケーションが Amazon SNS に登録されます。これにより、選択されたプラットフォーム用のプラットフォームアプリケーションオブジェクトが作成され、対応する PlatformApplicationArn が返されます。

プラットフォームエンドポイントを作成する

プッシュ通知サービスへのアプリケーションおよびモバイルデバイスの登録時に、プッシュ通知サービスはデバイストークンを返します。Amazon SNS はデバイストークンを使用して、モバイルエンドポイントを作成します。ここに直接プッシュ通知メッセージを送信できます。詳細については、「Amazon SNS ユーザー通知の前提条件」および「ユーザー通知プロセスの概要」を参照してください。

このセクションでは、プラットフォームエンドポイントの推奨作成方法について説明します。

トピック

- プラットフォームエンドポイントの作成
- 擬似コード
- AWS SDK の例
- トラブルシューティング

プラットフォームエンドポイントの作成

Amazon SNS を使用してアプリケーションにプッシュ通知を送信するには、まずプラットフォームエンドポイントの作成アクションを呼び出すことで、そのアプリケーションのデバイストークンを Amazon SNS に登録する必要があります。このアクションは、パラメーターとしてプラットフォームアプリケーションの Amazon リソースネーム (ARN) およびデバイストークンを使用し、作成されたプラットフォームエンドポイントの ARN を返します。

`CreatePlatformEndpoint`アクションは次のことを行います。

- プラットフォームエンドポイントが既に存在する場合、再度作成しないでください。呼び出し元に既存のプラットフォームエンドポイントの ARN を返します。
- デバイストークンが同じだが設定が異なるプラットフォームエンドポイントが既に存在する場合、再度作成しないでください。呼び出し元に例外をスローします。
- プラットフォームエンドポイントが存在しない場合は、作成します。呼び出し元に新しく作成したプラットフォームエンドポイントの ARN を返します。

このアプローチでは、作業エンドポイントが常に提供されるとは限らないため、アプリケーションが起動するたびにプラットフォームエンドポイントの作成アクションを呼び出さないでください。これは、例えばアプリケーションがアンインストールされて同じデバイスに再インストールされ、その

アプリケーションのエンドポイントが既に存在しているが無効な場合などに発生する可能性があります。登録プロセスに成功すると、以下のことが達成されます。

1. このアプリケーションデバイスの組み合わせにプラットフォームエンドポイントが存在することを確認します。
2. プラットフォームエンドポイントのデバイストークンが最新の有効なデバイストークンであることを確認します。
3. プラットフォームエンドポイントが有効であり、使用できる状態にあることを確認します。

擬似コード

次の擬似コードは、さまざまな開始条件において有効な最新の作業プラットフォームエンドポイントを作成するための推奨される方法について説明します。このアプローチは、アプリケーションが初めて登録されるかどうか、このアプリケーションのプラットフォームエンドポイントが既に存在するかどうか、プラットフォームエンドポイントが有効かどうか、適切なデバイストークンがあるかどうかなどに関係なく機能します。重複するプラットフォームエンドポイントが作成されたり、既に最新のプラットフォームエンドポイントが有効になっている場合に既存のプラットフォームエンドポイントが変更されたりしないため、連続して複数回呼び出しても安全です。

```
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
```

このアプローチは、アプリケーションが自身を登録または再登録するときいつでも使用できます。また、デバイストークンの変更について Amazon SNS に通知するときも使用できます。この場合、最新のデバイストークン値を持つアクションを呼び出すだけです。このアプローチについて注意が必要な点は次のとおりです。

- プラットフォームエンドポイントの作成アクションの呼び出しが考えられる 2 つのケースがあります。アプリケーションが自身のプラットフォームエンドポイント ARN を認識していない、ごく最初の時点で呼び出される可能性があります。これは初回登録時に発生します。さらに、最初のエンドポイント属性の取得アクションが not-found 例外で失敗した場合にも呼び出されます。これは、アプリケーションがエンドポイント ARN を認識しているが、削除された場合に発生する可能性があります。
- エンドポイント属性の取得アクションは、プラットフォームエンドポイントが作成されたばかりの場合でもプラットフォームエンドポイントの状態を確認するために呼び出されます。これは、プラットフォームエンドポイントが既に存在するが無効になっている場合に発生します。この場合、プラットフォームエンドポイントの作成アクションが成功しますが、プラットフォームエンドポイントは有効にならないため、成功を返す前にプラットフォームエンドポイントの状態をもう一度確認する必要があります。

AWS SDK の例

次のコードは、SDK が提供する Amazon SNS クライアントを使用して前述の疑似コードを実装する方法を示しています。 AWS

SDK を使用するには、 AWS 認証情報を使用して SDK を設定する必要があります。詳細については、「AWS SDK とツールのリファレンスガイド」の「共有設定ファイルと認証情報ファイル」を参照してください。

CLI

AWS CLI

プラットフォームアプリケーションのエンドポイントを作成するには

次の `create-platform-endpoint` の例では、指定したトークンを使用して、指定したプラットフォームアプリケーションのエンドポイントを作成します。

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
```

```
    --token EXAMPLE12345...
```

出力:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK for Java 2.x

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

詳細については、「モバイルプッシュ API アクション」を参照してください。

トラブルシューティング

古いデバイストークンを使用してプラットフォームエンドポイントの作成を繰り返し呼び出す

特に FCM エンドポイントの場合、アプリケーションが最初に発行されたデバイストークンを保存し、アプリケーションの起動時にそのデバイストークンを使用して create platform エンドポイントを呼び出すのが最善だと思うかもしれません。これは、アプリケーションがデバイストークンの状態を管理する必要がなくなり、Amazon SNS がサービストークンを最新の値に自動的に更新するため、適切に思えるかもしれません。しかし、この解決策には多くの深刻な問題があります。

- Amazon SNS は、有効期限切れのデバイストークンを新しいデバイストークンに更新するために FCM からのフィードバックが必要です。FCM は、古いデバイストークンの情報を当分の間保持しますが、無期限に保持するわけではありません。FCM が古いデバイストークンと新しいデバイストークンの関連性の認識を失うと、Amazon SNS はプラットフォームエンドポイントに保存されたデバイストークンを適切な値に更新することができなくなります。代わりに、プラットフォームエンドポイントが無効になるだけです。
- プラットフォームアプリケーションには、同じデバイストークンに対応する複数のプラットフォームエンドポイントが含められます。
- Amazon SNS では、同じデバイストークンを使用して作成できるプラットフォームエンドポイントの数に限度が適用されます。最終的に、新しいエンドポイントの作成は無効なパラメータの例外によって失敗し、「This endpoint is already registered with a different token.」 というエラーメッセージが表示されます。

FCM エンドポイントの管理について詳しくは、を参照してください。Firebase クラウドメッセージング (FCM) エンドポイント管理

無効なデバイストークンに関連付けられたプラットフォームエンドポイントを再度有効にする

モバイルプラットフォーム (APN や FCM など) が、発行リクエストで使用されたデバイストークンが無効であったことを Amazon SNS に通知すると、Amazon SNS はそのデバイストークンに関連付

けられたプラットフォームエンドポイントを無効にします。その後、Amazon SNS はそのデバイストークンへのその後の発行を拒否します。プラットフォームエンドポイントを再度に有効にして発行を継続すれば最適と思うかもしれませんが、ほとんどの場合これはうまくいきません。発行されるメッセージは配信されず、プラットフォームエンドポイントはその後まもなく無効になります。

これは、プラットフォームエンドポイントに関連付けられたデバイストークンが間違いなく無効であるためです。プラットフォームエンドポイントは、インストールされているどのアプリケーションにも応答しないため、配信が成功することはありません。次回発行されると、モバイルプラットフォームはデバイストークンが無効であることを Amazon SNS にもう一度通知し、Amazon SNS はプラットフォームエンドポイントをもう一度無効にします。

無効なプラットフォームエンドポイントを再度有効にするには、有効なデバイストークンに関連付けた後 (エンドポイント属性の設定アクションを呼び出して)、有効にする必要があります。その場合のみ、そのプラットフォームエンドポイントへの配信は正常に行われます。デバイストークンを更新しないでプラットフォームエンドポイントを有効にできるのは、そのエンドポイントに関連付けられたデバイストークンが無効であったが、再度有効となった場合のみです。これは、例えばアプリケーションがアンインストールされて同じモバイルデバイスに再インストールされ、同じデバイストークンを受け取った場合などに発生します。上に示したアプローチではこれが行われます。関連付けられたデバイストークンが使用可能な最新のものであると確認してから、プラットフォームエンドポイントを再度有効にすることのみ確実に行ってください。

デバイストークンまたは登録 ID を追加する

Apple Push Notification Service (APNs) や Firebase Cloud Messaging (FCM) などの通知サービスに初めてアプリやモバイルデバイスを登録すると、通知サービスからデバイストークンまたは登録 ID が通知サービスから返されます。デバイストークンまたは登録 ID を Amazon SNS に追加すると、それらは `PlatformApplicationArn` API とともに、アプリケーションやデバイスのエンドポイントを作成するために使用されます。Amazon SNS がエンドポイントを作成するときに、`EndpointArn` が返されます。`EndpointArn` は、通知メッセージの送信先のアプリやモバイルデバイスを Amazon SNS が知るための方法です。

次の方法を使用して、デバイストークンと登録 ID を Amazon SNS に追加できます。

- AWS Management Console を使用して手動で AWS に 1 つのトークンを追加する
- `CreatePlatformEndpoint` API を使用して複数のトークンをアップロードする
- 将来アプリケーションをインストールするデバイスからトークンを登録する

手動でデバイストークンまたは登録 ID を追加するには

1. Amazon SNS コンソールにサインインします。
2. [モバイル] を選択して [プッシュ通知] を選択します。
3. [プラットフォームアプリケーション] セクションでアプリケーションを選択し、[編集] を選択します。プラットフォームアプリケーションをまだ作成していない場合は、ここで作成します。これを行う手順については、「プラットフォームアプリケーションを作成する」を参照してください。
4. [エンドポイントの追加] を選択します。
5. [エンドトークン] ボックスに、通知サービスに応じてトークン ID または登録 ID を入力します。例えば、ADM や FCM では、登録 ID を入力します。
6. (オプション) [ユーザーデータ] ボックスに、エンドポイントに関連付ける任意の情報を入力します。Amazon SNS はこのデータは使用しません。データは UTF-8 形式で、2 KB 未満でなければなりません。
7. 最後に、[エンドポイントの追加] を選択します。

   これでエンドポイントを作成したので、直接モバイルデバイスにメッセージを送信したり、トピックにサブスクリプションしているモバイルデバイスにメッセージを送信したりできます。

`CreatePlatformEndpoint` API を使用して複数のトークンをアップロードするには

次の手順は、AWS に用意されているサンプルの Java アプリケーション (`bulkupload` パッケージ) を使用して複数のトークン (デバイストークンまたは登録 ID) を Amazon SNS にアップロードする方法を示しています。既存のトークンのアップロードを開始するには、このサンプルアプリを使用できます。

> **Note**
>
> 以下の手順では、Eclipse Java IDE を使用します。これらの手順では、AWS SDK for Java をインストールし、AWS アカウント の AWS セキュリティ認証情報があることを前提としています。詳細については、「AWS SDK for Java」を参照してください。認証情報の取得方法の詳細については、https://docs.aws.amazon.com/general/latest/gr/getting-aws-sec-creds.htmlの「AWS 全般のリファレンスセキュリティ認証情報の取得方法」を参照してください。

1. snsmobilepush.zip ファイルをダウンロードして解凍します。

2. Eclipse で新しい Java プロジェクトを作成します。
3. 新しく作成したされた Java プロジェクトの最上位ディレクトリに SNSSamples フォルダーをインポートします。Eclipse で、Java プロジェクトの名前を右クリックし、[Import] を選択して、[General] を展開します。次に、[File System]、[Next] の順に選択し、SNSSamples フォルダーを参照します。その後、[OK]、[Finish] の順に選択します。
4. OpenCSV library のコピーをダウンロードし、bulkupload パッケージのビルドパスに追加します。
5. bulkupload パッケージに含まれている BulkUpload.properties ファイルを開きます。
6. 以下を BulkUpload.properties に追加します。

   - エンドポイントを追加する ApplicationArn。
   - トークンを含む CSV ファイルの場所の絶対パス。
   - Amazon SNS が正しく解析するトークンと、解析に失敗するトークンのログ記録のために作成される CSV ファイルの名前 (goodTokens.csv、badTokens.csv など)。
   - (オプション) トークンを含む CSV ファイルで区切り記号と引用符を指定する文字。
   - (オプション) エンドポイントを同時に作成するために使用するスレッドの数。デフォルトは 1 スレッドです。

   完了した BulkUpload.properties は次のようになります。

   ```
   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. BatchCreatePlatformEndpointSample.java アプリケーションを実行してトークンを Amazon SNS にアップロードします。

   この例では、正常に Amazon SNS にアップロードされたトークン用に作成されたエンドポイントは goodTokens.csv に記録され、正しい形式でないトークンは badTokens.csv に記録されます。さらに、次のような内容を含む STD OUT ログが Eclipse のコンソールに書き込まれます。

```
<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
```

将来アプリをインストールするデバイスからトークンを登録するには

次の 2 つのオプションのうちの 1 つを使用できます。

- Use the Amazon Cognito service: モバイルアプリケーションでは、Amazon SNS プラットフォームアプリケーションと関連付けられた認証情報を作成するために、認証情報が必要です。一定期間後に有効期限が切れる一時認証情報を使用することをお勧めします。ほとんどの場合、Amazon Cognito を使用して一時的なセキュリティ認証情報を作成することをお勧めします。詳細については、「Amazon Cognito デベロッパーガイド」をご覧ください。アプリケーションが Amazon SNS に登録したときに通知を受信する場合は、新しいエンドポイント ARN を提供する Amazon SNS イベントを受信するように登録できます。`ListEndpointByPlatformApplication` API を使用して、Amazon SNS に登録されたエンドポイントの詳細な一覧を取得することもできます。
- [Use a proxy server]: 各インストールで呼び出し、登録するためにモバイルアプリケーション用にアプリケーションのインフラストラクチャがすでにセットアップされている場合は、引き続きこのセットアップを使用できます。サーバーはプロキシとして動作し、保存したいユーザーデータとともに、Amazon SNS モバイルプッシュ通知にデバイストークンを渡します。そのためには、プロキシサーバーは AWS の認証情報を使用して Amazon SNS に接続し、`CreatePlatformEndpoint` API コールを使用してトークン情報をアップロードします。新しく作成されたエンドポイント Amazon リソースネーム (ARN) が返され、サーバーは Amazon SNS にそれ以降に発行の呼び出しを行うためにこれを保存できます。

Apple 認証方法

アプリのデベロッパーであることを識別する情報を提供することで、Amazon SNS に iOS または macOS アプリへのプッシュ通知の送信を許可できます。認証するには、プラットフォームアプリケーションを作成するときにキーまたは証明書 のいずれかを提供します。どちらも Apple Developer アカウントから取得できます。

トークン署名キー

Amazon SNS が Apple Push Notification Service (APN) 認証トークンの署名に使用するプライベート署名キー。

署名キーを提供する場合、Amazon SNS はトークンを使用して、送信するプッシュ通知ごとに APN を使用して認証を行います。署名キーを使用すると、APN 本番環境とサンドボックス環境に通知を送信できます。

署名キーの有効期限は失効しないため、複数のアプリに対して同じ署名キーを使用できます。詳しくは、Apple ウェブサイトの デベロッパーアカウントヘルプ セクションの認証トークンを使用した APN との通信を参照してください。

証明書

プッシュ通知を送信するときに Amazon SNS が APN と認証するために使用する TLS 証明書です。証明書は Apple デベロッパーアカウントから取得できます。

証明書は 1 年後に失効します。この場合、新しい証明書を作成し、それを Amazon SNS に提供する必要があります。詳細については、Apple デベロッパーウェブサイトの「APN への証明書ベースの接続の確立」を参照してください。

AWS マネジメントコンソールを使用して APN の設定を管理するには

1. Amazon SNS コンソールにサインインします。
2. [モバイル] で、[プッシュ通知] を選択します。
3. APN 設定を編集するアプリケーションを選択し、[編集] を選択します。
4. 認証タイプの [編集] ページで、トークンまたは証明書のいずれかを選択します。
5. 証明書またはトークン署名キーの適切な認証情報をロードします。この情報は、Apple の開発者アカウントから取得できます。
6. 選択した認証タイプに応じて、次のいずれかの操作を行います。
   - [トークン] を選択した場合、Apple デベロッパーアカウントから以下の情報を提供してください。Amazon SNS は、認証トークンを作成するためにこの情報を必要とします。
     - 署名キー - .p8 ファイルとしてダウンロードした Apple デベロッパーアカウントの認証トークン署名キー。Apple では、署名キーは 1 回だけダウンロードすることができます。
     - 署名キー ID - 署名キーに割り当てられた ID。Amazon SNS は、認証トークンを作成するためにこの情報を必要とします。Apple デベロッパーアカウントでこの値を見つけるには、[証明書、ID およびプロファイル] を選択してから、[キー] セクションでキーを選択します。
     - チーム識別子 - Apple デベロッパーアカウントチームに割り当てられた ID。この値は、[メンバーシップ] ページで見つけられます。

- バンドル識別子 - アプリに割り当てられた ID。この値を見つけるには、[証明書、ID およびプロファイル] を選択し、[識別子] セクションで [App ID] を選択してから、アプリを選択します。

- [証明書] を選択した場合は、次の情報を入力します。
  - SSL 証明書 - TLS 証明書の「.p12」ファイル。Apple デベロッパーアカウントから証明書をダウンロードしてインストールした後で、このファイルを Keychain Access からエクスポートできます。
  - 証明書パスワード - 証明書にパスワードを割り当てた場合は、そのパスワードをここで指定します。

7. 変更が完了したら、[変更の保存] を選択します。

Firebase Cloud Messaging (FCM) の認証方法

このトピックでは、API AWS CLI およびで使用するために必要な FCM API (HTTP v1) 認証情報を Google から取得する方法について説明します。 AWS AWS Management Console

トピック

Important

2023 年 6 月 20 日 — Google は Firebase クラウドメッセージング (FCM) のレガシー HTTP API を廃止しました。Amazon SNS は FCM HTTP v1 API を使用するすべてのデバイスタイプへの配信をサポートするようになりました。中断を避けるため、2024 年 6 月 1 日またはそれ以前に既存のモバイルプッシュアプリケーションを最新の FCM HTTP v1 API に移行することをお勧めします。
2024 年 1 月 18 日 — Amazon SNS は Android デバイスへのモバイルプッシュ通知配信用の FCM HTTP v1 API のサポートを導入しました。
2024 年 3 月 26 日 — Amazon SNS は Apple デバイスと Webpush の送信先用の FCM HTTP v1 API をサポートしています。アプリケーションの中断を避けるため、2024 年 6 月 1 日ま

たはそれ以前に既存のモバイルプッシュアプリケーションを最新の FCM HTTP v1 API に移行することをお勧めします。

アプリケーションのデベロッパーであることを識別する情報を提供することで、Amazon SNS にアプリケーションへのプッシュ通知の送信を許可できます。認証するには、プラットフォームアプリケーションの作成時に API キーまたはトークンのいずれかを指定します。Firebase アプリケーションコンソールから次の情報を取得できます。

API キー

API キーは Firebase のレガシー API を呼び出すときに使用される認証情報です。FCM レガシー API は 2024 年 6 月 20 日に Google によって廃止されます。現在 API キーをプラットフォーム認証情報として使用している場合は、オプションとして [トークン] を選択し、Firebase アプリケーションに関連する JSON ファイルをアップロードすることで、プラットフォーム認証情報を更新できます。

Token

HTTP v1 API を呼び出す際には、有効期間の短いアクセストークンが使用されます。これは Firebase が推奨するプッシュ通知の送信用の API です。Firebase はアクセストークンを生成するために、プライベートキーファイル (service.json ファイルとも呼ばれます) の形式でデベロッパーに認証情報のセットを提供します。

前提条件

Amazon SNS で FCM の設定の管理を開始する前に、FCM service.json 認証情報を取得する必要があります。service.json 認証情報を取得するには、Google Firebase ドキュメントの「以前の HTTP から HTTP v1 に移行する」を参照してください。

FCM の設定の管理 (API)

API を使用して FCM プッシュ通知を作成できます。 AWS AWS アカウント内の Amazon SNS リソースの数とサイズには制限があります。詳細については、ガイドの「Amazon 簡易通知サービスのエンドポイントとクォータ」を参照してください。AWS 全般のリファレンス

Amazon SNS トピック (AWS API) と一緒に FCM プッシュ通知を作成するには

キー認証情報を使用する場合、`PlatformCredential` は `API key` です。トークン認証情報を使用する場合、`PlatformCredential` は JSON 形式のプライベートキーファイルです。

- CreatePlatformApplication

既存の Amazon SNS トピック (API) の FCM 認証情報タイプを取得するにはAWS

認証情報タイプ "AuthenticationMethod": "Token" または "AuthenticationMethod": "Key" を取得します。

- GetPlatformApplicationAttributes

既存の Amazon SNS トピックの FCM 属性を設定するには (AWS API)

FCM 属性を設定します。

- SetPlatformApplicationAttributes

FCM の設定の管理 (CLI)

AWS Command Line Interface (CLI) を使用して FCM プッシュ通知を作成できます。 AWS アカウント内の Amazon SNS リソースの数とサイズには制限があります。詳細については、「Amazon Simple Notification Service のエンドポイントとクォータ」を参照してください。

Amazon SNS トピックと共に FCM プッシュ通知を作成するには (AWS CLI)

キー認証情報を使用する場合、PlatformCredential は API key です。トークン認証情報を使用する場合、PlatformCredential は JSON 形式のプライベートキーファイルです。 AWS CLI を使用する場合、ファイルは文字列形式で、特殊文字は無視する必要があります。ファイルを正しくフォーマットするために、Amazon SNS では次のコマンドを使用することを推奨しています: SERVICE_JSON=`jq @json <<< cat service.json`:

- create-platform-application

既存の Amazon SNS トピックの FCM 認証情報タイプを取得するには (AWS CLI)

認証情報タイプ "AuthenticationMethod": "Token" または "AuthenticationMethod": "Key" を取得します。

- get-platform-application-attributes

既存の Amazon SNS トピックの FCM 属性を設定するには (AWS CLI)

FCM 属性を設定します。

- set-platform-application-attributes

FCM の設定の管理 (コンソール)

次の手順に従って、アプリケーションが FCM への接続に使用する認証情報を入力します。

1. Amazon SNS コンソールにサインインします。
2. [モバイル] で、[プッシュ通知] を選択します。
3. 既存の FCM アプリケーションを選択し、[編集] を選択します。プラットフォームアプリケーションをまだ作成していない場合は、「プラットフォームアプリケーションを作成する」を参照してください。
4. [編集] ページの [Firebase Cloud Messaging 認証情報] で、[トークン] または [キー] を選択します。次の情報は Firebase アプリケーションコンソールから取得できます。
   - [トークン] を選択した場合は、有効なプライベートキーファイルをアップロードします。このファイルの内容は、通知の送信時に有効期間の短いアクセストークンを生成するために使用されます。
   - [キー] を選択した場合は、Google API キーを入力します。
5. 変更が完了したら、[変更の保存] を選択します。

関連トピック

- Amazon SNS での Google Firebase クラウドメッセージング (FCM) v1 ペイロードの使用

Firebase クラウドメッセージング (FCM) エンドポイント管理

トピック

- デバイストークンの管理と保守
- 無効なトークンの検出
- 古いトークンの削除

デバイストークンの管理と保守

次の手順に従うことで、モバイルアプリケーションのプッシュ通知を確実に配信できます。

1. すべてのデバイストークン、対応する Amazon SNS エンドポイント ARN、タイムスタンプをアプリケーションサーバーに保存します。
2. 古いトークンをすべて削除し、対応する Amazon SNS エンドポイント ARN を削除します。

アプリを初めて起動すると、デバイスのデバイストークン (登録トークンとも呼ばれる) が届きます。このデバイストークンはデバイスのオペレーティングシステムによって作成され、FCM アプリケーションに関連付けられます。このデバイストークンを受け取ったら、プラットフォームエンドポイントとして Amazon SNS に登録できます。デバイストークン、Amazon SNS プラットフォームエンドポイント ARN、タイムスタンプは、アプリケーションサーバーまたは別の永続ストアに保存して保存することをお勧めします。デバイストークンを取得して保存するように FCM アプリケーションを設定するには、Google の Firebase ドキュメントの「登録トークンの取得と保存」を参照してください。

トークンを管理することは重要です。 up-to-date ユーザーのデバイストークンは、以下の条件下で変更される可能性があります。

1. モバイルアプリケーションが新しいデバイスに復元されます。
2. ユーザーがアプリケーションをアンインストールまたは更新します。
3. ユーザーはアプリケーションデータを消去します。

デバイストークンが変更された場合は、対応する Amazon SNS エンドポイントを新しいトークンで更新することをお勧めします。これにより、Amazon SNS は登録されたデバイスとの通信を継続できます。そのためには、モバイルアプリケーションに次の擬似コードを実装します。有効なプラットフォームエンドポイントを作成および管理するための推奨方法について説明しています。この方法は、モバイルアプリケーションが起動するたびに実行することも、バックグラウンドでスケジュールされたジョブとして実行することもできます。

擬似コード

次の FCM 疑似コードを使用して、デバイストークンの管理と管理を行います。

```
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
```

トークンの更新要件について詳しくは、Google の Firebase ドキュメントの「定期的なトークンの更新」をご覧ください。

無効なトークンの検出

無効なデバイストークンを含むメッセージが FCM v1 エンドポイントに送信されると、Amazon SNS は次の例外のいずれかを受け取ります。

- UNREGISTERED(HTTP 404) — Amazon SNS がこの例外を受け取ると、「FailureTypeof」の付いた配信失敗イベントを受け取ることになりInvalidPlatformToken、エンドポイントに関連付けられた *FailureMessage* of プラットフォームトークンが無効になります。この例外で配信が失敗すると、Amazon SNS はプラットフォームエンドポイントを無効にします。
- INVALID_ARGUMENT(HTTP 400) — Amazon SNS がこの例外を受け取った場合、デバイストークンまたはメッセージペイロードが無効であることを意味します。詳細については、Google ErrorCodeの Firebase ドキュメントを参照してください。

INVALID_ARGUMENTはどちらの場合でも返される可能性があるため、Amazon SNS は of を返しInvalidNotification、FailureType of の通知本文は無効です。FailureMessageこのエラーが表示されたら、ペイロードが正しいことを確認してください。正しければ、 up-to-dateデバイストークンが正しいことを確認してください。この例外で配信が失敗しても、Amazon SNS はプラットフォームエンドポイントを無効にしません。

InvalidPlatformToken配信失敗イベントが発生するもう 1 つのケースは、登録されたデバイストークンがそのメッセージを送信しようとしているアプリケーションのものではない場合です。この場合、Google は SENDER_ID_MISMATCH エラーを返します。この例外で配信が失敗すると、Amazon SNS はプラットフォームエンドポイントを無効にします。

FCM v1 API から受信したすべてのエラーコードは、 CloudWatchアプリケーションの配信ステータスロギングを設定する際に利用できます。

アプリケーションの配信イベントを受信するには、を参照してください。使用可能なアプリケーションイベント

古いトークンの削除

エンドポイントデバイスへのメッセージ配信が失敗し始めると、トークンは古いものとみなされます。Amazon SNS は、これらの古いトークンをプラットフォームアプリケーションの無効なエンドポイントとして設定します。無効になっているエンドポイントに公開すると、Amazon SNS は of `EventDeliveryFailure` のイベントを返し`EndpointDisabled`、`FailureType` `FailureMessage` of エンドポイントは無効になります。アプリケーションの配信イベントを受信するには、を参照してください使用可能なアプリケーションイベント。

Amazon SNS からこのエラーを受け取ったら、プラットフォームアプリケーションの古いトークンを削除または更新する必要があります。

モバイルプッシュ通知を送信する

このセクションでは、モバイルプッシュ通知を送信する方法について説明します。

トピック

- トピックへ発行する
- モバイルデバイスへ発行する
- プラットフォーム固有のペイロードによる公開

トピックへ発行する

Amazon SNS を使用して、トピックにサブスクリプションしたモバイルエンドポイントにメッセージを送信することもできます。このコンセプトは、「Amazon SNS とは」に説明されているように、Amazon SQS、HTTP/S、E メール、SMS などその他のエンドポイントタイプをトピックにサブスクライブするのと同じです。違いは、Amazon SNS は Apple Push Notification Service (APNS) や Google Firebase Cloud Messaging (FCM) などの通知サービスを通じて通信することです。通知サービスを通じて、サブスクライブしたモバイルエンドポイントがトピックに送信された通知を受信します。

モバイルデバイスへ発行する

モバイルデバイスのアプリケーションを表すエンドポイントに Amazon SNS プッシュ通知メッセージを直接送信できます。

直接メッセージを送信するには

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[Push notifications] (プッシュ通知) を選択します。
3. モバイルプッシュ通知ページの「プラットフォームアプリケーション」セクションで、たとえばアプリケーションの名前を選択します***MyApp***。
4. ***MyApp***ページの「エンドポイント」セクションでエンドポイントを選択し、「メッセージを公開」を選択します。
5. [エンドポイントへのメッセージの発行] ページで、モバイルデバイスのアプリケーションに表示されるメッセージを入力し、[メッセージの発行] を選択します。

   Amazon SNS によって、プラットフォーム通知サービスに通知メッセージが送信され、プラットフォーム通知サービスはアプリケーションにメッセージを送信します。

プラットフォーム固有のペイロードによる公開

AWS Management Console または Amazon SNS API を使用して、プラットフォーム固有のペイロードを含むカスタムメッセージをモバイルデバイスに送信できます。Amazon SNS API の使用については、モバイルプッシュ API アクション および `snsmobilepush.zip` の `SNSMobilePush.java` ファイルを参照してください。

トピック

- JSON 形式のメッセージの送信
- プラットフォーム固有のメッセージを送信する
- 複数プラットフォーム上のアプリケーションへのメッセージ送信
- アラートまたはバックグラウンド通知としてのメッセージを APN に送信する
- Amazon SNS での Google Firebase クラウドメッセージング (FCM) v1 ペイロードの使用

JSON 形式のメッセージの送信

プラットフォーム固有のペイロードを送信する場合、データは引用符をエスケープした、JSON キーと値のペア文字列である必要があります。

次の例では、FCM プラットフォーム用のカスタムメッセージを示します。

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
}
```

プラットフォーム固有のメッセージを送信する

カスタムデータをキーと値のペアとして送信することに加えて、プラットフォーム固有のキーと値のペアを送信できます。

次の例では、FCM `data` パラメータのカスタムデータのキーと値のペアの後に、FCM パラメータ `time_to_live` および `collapse_key` を含めています。

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Amazon SNS でサポートされている各プッシュ通知サービスでサポートされているキーと値のペアのリストについては次を参照してください。

> **Important**
>
> Amazon SNS は、Android デバイスにモバイルプッシュ通知を送信するための Firebase クラウドメッセージング (FCM) HTTP v1 API をサポートするようになりました。
> 2024 年 3 月 26 日 — Amazon SNS は Apple デバイスと Webpush の送信先用の FCM HTTP v1 API をサポートしています。アプリケーションの中断を避けるため、2024 年 6 月 1 日またはそれ以前に既存のモバイルプッシュアプリケーションを最新の FCM HTTP v1 API に移行することをお勧めします。

- APN ドキュメントのペイロードキーのリファレンス

- FCM ドキュメントのFirebase Cloud Messaging HTTP プロトコル
- ADM ドキュメントのメッセージの送信

複数プラットフォーム上のアプリケーションへのメッセージ送信

FCM や APN など、複数プラットフォーム向けのデバイスにインストールされたアプリケーションにメッセージを送信するには、Amazon SNS のトピックにモバイルエンドポイントをサブスクリプションしてから、トピックにメッセージを発行します。

次の例は、APN、FCM、および ADM でサブスクリプションしたモバイルエンドポイントに送信するメッセージを示しています。

```
{
  "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\"}}"
}
```

アラートまたはバックグラウンド通知としてのメッセージを APN に送信する

Amazon SNS は APN にメッセージを `alert` または `background` 通知 (詳細については、APN のドキュメントの「バックグラウンド更新をアプリケーションにプッシュする」を参照してください)。

- `alert` APN 通知では、ユーザーに通知する方法として、アラートメッセージを表示するか、サウンドを鳴らすか、アプリケーションのアイコンにバッジを追加します。
- `background` APN 通知は、ユーザーに通知することなく、ウェイクアップするか、通知の内容に基づいて動作するようにアプリケーションに指示します。

APN ヘッダーのカスタム値を指定する

Amazon SNS `Publish` API アクション、 AWS SDK、またはを使用して、`AWS.SNS.MOBILE.APNS.PUSH_TYPE`リザーブドメッセージ属性のカスタム値を指定すること

をお勧めします。 AWS CLI次の CLI の例では、指定したトピックの content-available を 1 に設定し、apns-push-type を background に設定します。

```
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
```

ペイロードからの APN プッシュタイプヘッダーを推測する

apns-push-type APNヘッダーを設定しないと、JSON 形式の APN ペイロードに設定されている aps ディクショナリ内の content-available キーに応じて、Amazon SNS がヘッダーを alert または background に設定します。

> **Note**
>
> apns-push-type ヘッダーは他の値に設定できますが、Amazon SNS が推測できるのは alert ヘッダーまたは background ヘッダーのみです。

- apns-push-type は、alert に設定されます。
  - aps ディクショナリで content-available が 1 に設定されていて、ユーザーの操作をトリガーする 1 つ以上のキーが含まれている場合。
  - aps ディクショナリで content-available が 0 に設定されているか、または content-available キーが存在しない場合。
  - content-available キーの値が整数またはブール値でない場合。
- apns-push-type は、background に設定されます。
  - aps ディクショナリに 1 に設定された content-available のみが含まれ、ユーザーの操作をトリガーする他のキーが含まれていない場合。

> **Important**
>
> Amazon SNSが APN の raw 設定オブジェクトをバックグラウンドのみの通知として送信する場合は、`content-available` を 1 に設定して `aps` ディクショナリに含める必要があります。カスタムキーを含めることもできますが、ユーザー操作をトリガーするキー (アラート、バッジ、サウンドなど) を `aps` ディクショナリに含めることはできません。

raw 設定オブジェクトの例を次に示します。

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

この例の場合、Amazon SNS はメッセージの `apns-push-type` APN ヘッダーを `background` に設定します。Amazon SNS が `apn` ディクショナリが 1 に設定された `content-available` キーを含み、ユーザー操作をトリガーする他のキーを含まないことを検出すると、Amazon SNS はヘッダーを `background` に設定します。

Amazon SNS での Google Firebase クラウドメッセージング (FCM) v1 ペイロードの使用

Amazon SNS は、FCM HTTP v1 API を使用して、アンドロイド、iOS、およびウェブプッシュの宛先に通知を送信することをサポートしています。このトピックでは、CLI または Amazon SNS API を使用してモバイルプッシュ通知を発行するときのペイロード構造の例を紹介します。

FCM 通知を送信する際には、ペイロードに次のメッセージタイプを含めることができます。

- データメッセージ — データメッセージはクライアントアプリによって処理され、カスタムのキーと値のペアが含まれます。データメッセージを作成するときは、JSON `data` オブジェクトを含むキーを値として含め、カスタムのキーと値のペアを入力する必要があります。
- 通知メッセージまたは表示メッセージ — 通知メッセージには、FCM SDK によって処理される定義済みのキーセットが含まれます。これらのキーは、配信先のデバイスのタイプによって異なります。プラットフォーム固有の通知キーについて詳しくは、以下を参照してください。
  - Android の通知キー
  - APNS 通知キー
  - ウェブプッシュ-通知キー

FCM メッセージタイプについて詳しくは、Google の Firebase ドキュメントの「メッセージタイプ」を参照してください。

目次

FCM v1 のペイロード構造を使用してメッセージを送信する

FCM アプリケーションを初めて作成する場合や FCM v1 の機能を利用したい場合は、FCM v1 フォーマットのペイロードを送信するようにオプトインできます。そのためには、最上位のキーを含める必要があります。`fcmV1Message`FCM v1 ペイロードの作成について詳しくは、Google の Firebase ドキュメントの「従来の FCM API から HTTP v1 への移行」と「プラットフォーム間でのメッセージのカスタマイズ」を参照してください。

Amazon SNS に送信された FCM v1 のペイロードの例:

> **Note**
>
> GCM次の例で使用されているキー値は、Amazon SNS を使用して通知を発行するときに String としてエンコードする必要があります。

```
{
  "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\",
                }
            }
        }
    }
  }"
}
```

JSON ペイロードを送信するときは、message-structure必ずその属性をリクエストに含め、に設定してください。json

CLI の例:

```
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
```

FCM v1 形式のペイロードの送信について詳しくは、Google の Firebase ドキュメントの以下を参照してください。

- 従来の FCM API から HTTP v1 に移行してください。
- FCM メッセージについて
- REST リソース:projects.messages

従来のペイロード構造を使用して FCM v1 API にメッセージを送信します。

FCM v1 に移行する場合、従来の認証情報に使用していたペイロード構造を変更する必要はありません。Amazon SNS はペイロードを新しい FCM v1 ペイロード構造に変換し、Google に送信します。

入力メッセージペイロード形式:

```
{
  "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\"}}"
}
```

Google に送信されたメッセージ:

```
{
  "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"
    }
  }
}
```

潜在的なリスク

- レガシーから v1 へのマッピングは Apple プッシュ通知サービス (APNS) headers やキーをサポートしていません。fcm_optionsこれらのフィールドを使用する場合は、FCM v1 ペイロードを送信してください。
- FCM v1 では、APNs デバイスにサイレント通知を送信するためにメッセージヘッダーが必要な場合があります。現在 APNs デバイスにサイレント通知を送信している場合、従来の方法では動作

しません。予期しない問題を避けるため、代わりに FCM v1 ペイロードを使用することをお勧めします。APN ヘッダーの一覧と用途については、Apple 開発者ガイドの「APN との通信」を参照してください。

- 通知の送信時に TTL Amazon SNS 属性を使用している場合は、androidフィールドでのみ更新されます。TTLAPNS 属性を設定する場合は、FCM v1 ペイロードを使用してください。
- android、apns、webpushおよびキーはマッピングされ、指定されたすべての関連キーが入力されます。たとえば、3 title つのプラットフォームすべてで共有されるキーを指定すると、FCM v1 マッピングにより、3 つのプラットフォームすべてに指定したタイトルが入力されます。
- プラットフォーム間で共有されるキーの中には、異なる値タイプを想定しているものがあります。たとえば、badgeapnsに渡されるキーは整数値を必要とし、badgewebpushに渡されるキーは文字列値を期待します。badgeキーを指定した場合、FCM v1 マッピングでは、有効な値を指定したキーのみが入力されます。

FCM 配信失敗イベント

次の表は、FCM v1 通知リクエストについて Google から受け取ったエラー/ステータスコードに対応する Amazon SNS 障害タイプを示しています。FCM v1 API から受信したすべてのエラーコードは、 CloudWatch アプリケーションの配信ステータスロギングを設定する際に利用できます。

FCM エラー/ステータスコード	Amazon SNS 障害タイプ	障害メッセージ	原因と緩和策
UNREGISTERED	InvalidPlatformToken	エンドポイントに関連付けられたプラットフォームトークンは無効です。	エンドポイントに添付されているデバイストークンが古くなっているか無効です。Amazon SNS によってエンドポイントが無効になりました。Amazon SNS エンドポイントを最新のデバイストークンに更新します。
INVALID_ARGUMENT	InvalidNotification	通知本文は無効です。	デバイストークンまたはメッセージペイ

FCM エラー/ステータスコード	Amazon SNS 障害タイプ	障害メッセージ	原因と緩和策
			ロードが無効である可能性があります。メッセージペイロードが有効であることを確認してください。メッセージペイロードが有効な場合は、Amazon SNS エンドポイントを最新のデバイストークンに更新します。
SENDER_ID_MISMATCH	InvalidPlatformToken	エンドポイントに関連付けられているプラットフォームトークンは無効です。	デバイストークンに関連付けられているプラットフォームアプリケーションには、デバイストークンに送信する権限がありません。Amazon SNS プラットフォームアプリケーションで正しい FCM 認証情報を使用していることを確認します。

FCM エラー/ステータスコード	Amazon SNS 障害タイプ	障害メッセージ	原因と緩和策
UNAVAILABLE	DependencyUnavailable	依存関係は利用できません。	FCM はリクエストを時間内に処理できませんでした。Amazon SNS によって実行されたすべての再試行が失敗しました。これらのメッセージはデッドレターキュー (DLQ) に保存し、後で再ドライブできます。
INTERNAL	UnexpectedFailure	予期しない障害です。Amazon にお問い合わせください。失敗フレーズ [内部エラー]。	FCM サーバーは、リクエストを処理しようとしたときにエラーが発生しました。Amazon SNS によって実行されたすべての再試行が失敗しました。これらのメッセージはデッドレターキュー (DLQ) に保存し、後で再ドライブできます。
THIRD_PARTY_AUTH_ERROR	InvalidCredentials	プラットフォームアプリケーションの認証情報は無効です。	iOS デバイスまたは Webpush デバイスを対象としたメッセージを送信できませんでした。開発用と本番用の認証情報が有効であることを確認してください。

FCM エラー/ステータスコード	Amazon SNS 障害タイプ	障害メッセージ	原因と緩和策
QUOTA_EXCEEDED	Throttled	[gcm] によってリクエストが調整されました。	メッセージレートクォータ、デバイスメッセージレートクォータ、またはトピックメッセージレートクォータを超えました。この問題を解決する方法については、Google ErrorCodeの Firebase ドキュメントのを参照してください。
PERMISSION_DENIED	InvalidNotification	通知本文は無効です。	PERMISSION_DENIED 例外が発生した場合、呼び出し元 (FCM アプリケーション) には、ペイロード内の指定された操作を実行する権限がありません。FCM コンソールに移動し、認証情報で必要な API アクションが有効になっていることを確認します。

モバイルアプリケーションの属性

Amazon Simple Notification Service (Amazon SNS) では、プッシュ通知メッセージの配信ステータスの記録がサポートされています。アプリケーション属性を設定した後、Amazon SNS からモバイルエンドポイントに送信されたメッセージのログエントリが CloudWatch Logs に送信されます。

メッセージの配信ステータスを記録することは、以下のように運用をよりよく把握するのに役立ちます。

- プッシュ通知メッセージが Amazon SNS からプッシュ通知サービスに配信されたかどうかがわかります。
- プッシュ通知サービスから Amazon SNS に送信されたレスポンスを特定します。
- メッセージのドウェル時間 (発行のタイムスタンプからプッシュ通知サービスへの配信直前までの時間) を決定します。

メッセージの配信ステータスのアプリケーション属性を設定するには、AWS Management Console、AWS Software Development Kit (SDK)、またはクエリ API を使用できます。

トピック

- AWS Management Console を使用してメッセージの配信ステータスの属性を設定する
- Amazon SNS メッセージ配信ステータス CloudWatch ログの例
- AWS SDK でメッセージの配信ステータスの属性を設定する
- プラットフォームのレスポンスコード

AWS Management Console を使用してメッセージの配信ステータスの属性を設定する

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで [モバイル]、[プッシュ通知] を選択します。
3. [プラットフォームアプリケーション]セクションで、CloudWatch Logs を受け取るエンドポイントを含むアプリケーションを選択します。
4. [アプリケーションアクション]、[配信ステータス] の順に選択します。
5. [配信ステータス] ダイアログボックスで、[IAM ロールの作成] を選択します。

   IAM コンソールにリダイレクトされます。
6. [許可] を選択して、お客様に代わって CloudWatch Logs を使用する書き込みアクセス許可を Amazon SNS に付与します。
7. ここで、[配信ステータス] ダイアログボックスに戻り、[Percentage of Success to Sample (0-100)] フィールドに、CloudWatch Logs を受信するために送信される正常なメッセージの割合を数字で入力します。

> **Note**
>
> メッセージの配信ステータスのアプリケーション属性を設定した後は、メッセージの配信に失敗すると、必ず CloudWatch Logs が生成されます。

8. 最後に、[設定の保存] を選択します。これで、メッセージの配信ステータスを含む CloudWatch Logs を参照して解析できます。CloudWatch の使用方法の詳細については、「CloudWatch のドキュメント」を参照してください。

Amazon SNS メッセージ配信ステータス CloudWatch ログの例

アプリケーションエンドポイントのメッセージの配信ステータスの属性を設定した後は、CloudWatch Logs が生成されます。以下は、JSON 形式のログの例です。

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"
  }
}
```

プッシュ通知サービスのレスポンスコードの一覧は、「プラットフォームのレスポンスコード」を参照してください。

AWS SDK でメッセージの配信ステータスの属性を設定する

AWS SDK には、Amazon SNS でメッセージの配信ステータスの属性を使用するための API がいくつかの言語で用意されています。

次の Java の例は、SetPlatformApplicationAttributes API を使用して、プッシュ通知メッセージの配信ステータスのアプリケーション属性を設定する方法を示しています。メッセージの配信ステータスには、SuccessFeedbackRoleArn、FailureFeedbackRoleArn、および SuccessFeedbackSampleRate の属性を使用できます。SuccessFeedbackRoleArn および FailureFeedbackRoleArn 属性は、お客様に代わって CloudWatch Logs を使用する書き込みアクセス許可を Amazon SNS に付与するために使用します。SuccessFeedbackSampleRate 属性は、正常な配信メッセージのサンプルレートの割合 (0〜100) を指定するためのものです。FailureFeedbackRoleArn 属性を設定した後は、メッセージの配信に失敗すると、必ず CloudWatch Logs が生成されます。

```
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);
```

Java 用 SDL の詳細については、「AWS SDK for Java の開始方法」を参照してください。

プラットフォームのレスポンスコード

以下は、プッシュ通知サービスのレスポンスコードへのリンクの一覧です。

プッシュ通知サービス	レスポンスコード
Amazon Device Messaging (ADM)	ADM のドキュメント の「レスポンスの形式」を参照してください。
Apple Push Notification Service (APNS)	『ローカルおよびリモート通知プログラミングガイド』の「APN との通信におけるAPN からの HTTP/2 レスポンス」を参照してください。
Firebase Cloud Messaging (FCM)	Firebase Cloud Messaging ドキュメントの「Downstream Message Error Response Codes」を参照してください。
Windows Phone (MPNS) 用 Microsoft Push Notification Service (MPNS)	Windows 8 開発ドキュメントの「Push Notification Service Response Codes for Windows Phone 8」を参照してください。
Windows Push Notification Services (WNS)	Windows 8 開発ドキュメントで「Push Notification Service Request and Response Headers (Windows Runtime Apps)」の「Response codes」を参照してください。

モバイルアプリケーションのイベント

Amazon SNS は、特定のアプリケーションイベントが発生する際のトリガー通知をサポートします。その場合、そのイベントでプログラムによるいくつかのアクションを実行することができます。アプリケーションには、Apple Push Notification Service (APN)、Firebase Cloud Messaging (FCM)、Windows Push Notification Service (WNS) などのプッシュ通知サービスのサポートが含まれ

ている必要があります。アプリケーションイベント通知は、Amazon SNS コンソール AWS CLI、または AWS SDKs。

トピック

- 使用可能なアプリケーションイベント
- モバイルプッシュ通知を送信する

使用可能なアプリケーションイベント

アプリケーションイベント通知は、個々のプラットフォームエンドポイントがいつ作成、削除、更新、配信エラーとなったかを追跡します。アプリケーションイベントの属性名は次のとおりです。

属性名	通知トリガー
`EventEndpointCreated`	新しいエンドポイントがアプリケーションに追加されます。
`EventEndpointDeleted`	アプリケーションに関連付けられたすべてのプラットフォームエンドポイントが削除されます。
`EventEndpointUpdated`	アプリケーションに関連付けられたプラットフォームエンドポイントのすべての属性が変更されます。
`EventDeliveryFailure`	アプリケーションに関連付けられた任意のプラットフォームエンドポイントへの配信は、永続的にエラーとなります。 **Note** プラットフォームアプリケーション側で配信エラーを追跡するには、アプリケーションのメッセージ配信ステータスイベントにサブスクライブします。詳細については、「メッセージの配信ステータスの Amazon SNS アプリケーション属性を使用する」を参照してください。

アプリケーションには任意の属性を関連付けることができ、これでそのイベント通知を受け取れるようになります。

モバイルプッシュ通知を送信する

アプリケーションイベント通知を送信するには、各タイプのイベントの通知を受信できるよう、トピックを指定します。Amazon SNS が通知を送信する場合、トピックは、プログラムによるアクションを取るエンドポイントにそれらをルーティングすることができます。

> **Important**
>
> 大容量アプリケーションは多数のアプリケーションイベント通知 (例えば、数万回) を作成するので、E メールアドレス、電話番号、モバイルアプリケーションなど、人間が使用するためのエンドポイントに負荷をかけます。アプリケーションイベント通知をトピックに送信する場合は、以下のガイドラインを検討してください。
>
> - 通知を受け取る各トピックには、HTTP または HTTPS エンドポイント、Amazon SQS キュー、 AWS Lambda 関数など、プログラムによるエンドポイントのサブスクリプションのみを含める必要があります。
> - 通知によってトリガーされる処理量を減らすために、各トピックのサブスクリプションを少数 (例えば、5 以下) に制限します。

アプリケーションイベント通知は、Amazon SNS コンソール、 AWS Command Line Interface （ AWS CLI ）、または AWS SDKsを使用して送信できます。

AWS Management Console

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで [モバイル]、[プッシュ通知] を選択します。
3. モバイルプッシュ通知ページのプラットフォームアプリケーションセクションで、アプリケーションを選択し、編集を選択します。
4. [イベント通知] セクションを展開します。
5. [アクション]、[イベントの設定] を選択します。
6. 次のイベントで使用される ARN を入力します。

   - 作成されたエンドポイント
   - 削除されたエンドポイント
   - 更新されたエンドポイント
   - 配信失敗

7. [変更を保存] をクリックします。

AWS CLI

set-platform-application-attributes コマンドを実行します。

次の例では、4 つのアプリケーションすべてに対して同じ Amazon SNS トピックを設定します。

```
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

AWS SDK を使用して Amazon SNS API でSetPlatformApplicationAttributesリクエストを送信して、アプリケーションイベント通知を設定します。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、開始方法や以前のバージョンに関する情報など、「」を参照してくださいAWS SDK での Amazon SNS の使用。

モバイルプッシュ API アクション

Amazon SNS モバイルプッシュ API を使用するには、まず、プッシュ通知サービス (Apple Push Notification (APN)、Firebase Cloud Messaging (FCM) など) の前提条件を満たす必要があります。前提条件の詳細については、「Amazon SNS ユーザー通知の前提条件」を参照してください。

API を使用してモバイルアプリケーションおよびデバイスにプッシュ通知メッセージを送信するには、まず、CreatePlatformApplication アクションを使用する必要があります。このアクションは、PlatformApplicationArn 属性を返します。この PlatformApplicationArn 属性は、CreatePlatformEndpoint によって使用され、EndpointArn 属性が返されます。この EndpointArn 属性と Publish アクションを使用して、モバイルアプリケーションやデバイスに通知メッセージを送信することができます。また、EndpointArn 属性と Subscribe アクションを

使って、トピックへのサブスクリプションを行うこともできます。詳細については、「ユーザー通知プロセスの概要」を参照してください。

Amazon SNS モバイルプッシュ API は次のとおりです。

CreatePlatformApplication

デバイスやモバイルアプリケーションを登録できる、サポートされているいずれかのプッシュ通知サービス (例: APNS、GCM) のプラットフォームアプリケーションオブジェクトを作成します。PlatformApplicationArn 属性を返します。この属性は、CreatePlatformEndpoint アクションで使用されます。

CreatePlatformEndpoint

サポートされているプッシュ通知サービスのいずれかでデバイスおよびモバイルアプリケーションのエンドポイントを作成します。CreatePlatformEndpoint は、CreatePlatformApplication アクションから返される PlatformApplicationArn属性を使用します。CreatePlatformEndpoint を使用したときに返される EndpointArn 属性は、モバイルアプリケーションとデバイスに通知メッセージを送信するために Publish アクションで使用されます。

CreateTopic

メッセージが発行されるトピックを作成します。

DeleteEndpoint

サポートされているプッシュ通知サービスのいずれかでデバイスおよびモバイルアプリケーションのエンドポイントを削除します。

DeletePlatformApplication

プラットフォームアプリケーションオブジェクトを削除します。

DeleteTopic

トピックとすべてのサブスクリプションを削除します。

GetEndpointAttributes

デバイスおよびモバイルアプリケーションのエンドポイント属性を取得します。

GetPlatformApplicationAttributes

プラットフォームアプリケーションオブジェクトの属性を取得します。

`ListEndpointsByPlatformApplication`

サポートされているプッシュ通知サービスでのデバイスおよびモバイルアプリケーションのエンドポイントとエンドポイント属性を一覧表示します。

`ListPlatformApplications`

サポートされているプッシュ通知サービス用のプラットフォームアプリケーションオブジェクトを一覧表示します。

`Publish`

トピックのサブスクライブされたエンドポイントすべてに通知メッセージを送信します。

`SetEndpointAttributes`

デバイスおよびモバイルアプリケーションのエンドポイントの属性を設定します。

`SetPlatformApplicationAttributes`

プラットフォームアプリケーションオブジェクトの属性を設定します。

`Subscribe`

エンドポイントに確認メッセージを送信して、エンドポイントのサブスクライブに備えます。実際にサブスクリプションを作成するには、エンドポイントの所有者は確認メッセージからトークンと共に ConfirmSubscription アクションを呼び出す必要があります。

`Unsubscribe`

サブスクリプションを削除します。

モバイルプッシュ API エラー

Amazon SNS API から返されるモバイルプッシュに関するエラーを、次の表に示します。モバイルプッシュ用 Amazon SNS API の詳細については、「モバイルプッシュ API アクション」を参照してください。

エラー	説明	HTTPS ステータスコード	API アクション
Application Name is null string	必須のアプリケーション名が null に設定されています。	400	`CreatePlatformApplication`

エラー	説明	HTTPS ステータスコード	API アクション
Platform Name is null string	必須のプラットフォーム名が null に設定されています。	400	`CreatePlatformApplication`
Platform Name is invalid	プラットフォーム名に無効な値または範囲外の値が入力されました。	400	`CreatePlatformApplication`
APN - Principal is not a valid certificate	APN プリンシパル (SSL 証明書) に無効な証明書が指定されました。詳細については、Amazon Simple Notification Service API リファレンスの「CreatePlatformApplication」を参照してください。	400	`CreatePlatformApplication`
APN - Principal is a valid cert but not in a .pem format	SSL プリンシパル (SSL 証明書) に、.pem 形式ではない有効な証明書が指定されました。	400	`CreatePlatformApplication`
APNs — Principal is an expired certificate	APN プリンシパル (SSL 証明書) に失効した証明書が指定されました。	400	`CreatePlatformApplication`

エラー	説明	HTTPS ステータスコード	API アクション
APNs — Principal is not an Apple issued certificate	APN プリンシパル (SSL 証明書) に Apple によって発行されたものではない証明書が指定されました。	400	`CreatePlatformApplication`
APNs — Principal is not provided	APN プリンシパル (SSL 証明書) が指定されませんでした。	400	`CreatePlatformApplication`
APNs — Credential is not provided	APN 認証情報 (プライベートキー) が指定されませんでした。詳細については、Amazon Simple Notification Service API リファレンスの「CreatePlatformApplication」を参照してください。	400	`CreatePlatformApplication`
APNs — Credential are not in a valid .pem format	APN 認証情報 (プライベートキー) が有効な .pem 形式ではありません。	400	`CreatePlatformApplication`

エラー	説明	HTTPS ステータスコード	API アクション
FCM — serverAPIKey is not provided	FCM 認証情報 (API キー) が指定されませんでした。詳細については、Amazon Simple Notification Service API リファレンスの「CreatePlatformApplication」を参照してください。	400	`CreatePlatformApplication`
FCM — serverAPIKey is empty	FCM 認証情報 (API キー) が空です。	400	`CreatePlatformApplication`
FCM — serverAPIKey is a null string	FCM 認証情報 (API キー) が null です。	400	`CreatePlatformApplication`
FCM — serverAPIKey is invalid	API 認証情報 (API キー) が無効です。	400	`CreatePlatformApplication`
ADM — clientsecret is not provided	必須のクライアントシークレットが提供されていません。	400	`CreatePlatformApplication`
ADM — clientsecret is a null string	クライアントシークレットの必須文字列が null です。	400	`CreatePlatformApplication`
ADM — client_secret is empty string	クライアントシークレットの必須文字列が空です。	400	`CreatePlatformApplication`

エラー	説明	HTTPS ステータスコード	API アクション
ADM — client_secret is not valid	クライアントシークレットの必須文字列が無効です。	400	CreatePlatformApplication
ADM — client_id is empty string	クライアント ID の必須文字列が空です。	400	CreatePlatformApplication
ADM — clientId is not provided	クライアント ID の必須文字列が提供されていません。	400	CreatePlatformApplication
ADM — clientid is a null string	クライアント ID の必須文字列が null です。	400	CreatePlatformApplication
ADM — client_id is not valid	クライアント ID の必須文字列が無効です。	400	CreatePlatformApplication
EventEndpointCreated has invalid ARN format	EventEndpointCreated が無効な ARN 形式です。	400	CreatePlatformApplication
EventEndpointDeleted has invalid ARN format	EventEndpointDeleted が無効な ARN 形式です。	400	CreatePlatformApplication
EventEndpointUpdated has invalid ARN format	EventEndpointUpdated が無効な ARN 形式です。	400	CreatePlatformApplication
EventDeliveryAttemptFailure has invalid ARN format	EventDeliveryAttemptFailure が無効な ARN 形式です。	400	CreatePlatformApplication

エラー	説明	HTTPS ステータスコード	API アクション
EventDeliveryFailure has invalid ARN format	EventDeliveryFailure が無効な ARN 形式です。	400	CreatePlatformApplication
EventEndpointCreated is not an existing Topic	EventEndpointCreated は既存のトピックではありません。	400	CreatePlatformApplication
EventEndpointDeleted is not an existing Topic	EventEndpointDeleted は既存のトピックではありません。	400	CreatePlatformApplication
EventEndpointUpdated is not an existing Topic	EventEndpointUpdated は既存のトピックではありません。	400	CreatePlatformApplication
EventDeliveryAttemptFailure is not an existing Topic	EventDeliveryAttemptFailure は既存のトピックではありません。	400	CreatePlatformApplication
EventDeliveryFailure is not an existing Topic	EventDeliveryFailure は既存のトピックではありません。	400	CreatePlatformApplication
Platform ARN is invalid	プラットフォーム ARN が無効です。	400	SetPlatformAttributes
Platform ARN is valid but does not belong to the user	プラットフォーム ARN は有効ですが、このユーザーに属していません。	400	SetPlatformAttributes

エラー	説明	HTTPS ステータスコード	API アクション
APN - Principal is not a valid certificate	APN プリンシパル (SSL 証明書) に無効な証明書が指定されました。詳細については、Amazon Simple Notification Service API リファレンスの「CreatePlatformApplication」を参照してください。	400	SetPlatformAttributes
APN - Principal is a valid cert but not in a .pem format	SSL プリンシパル (SSL 証明書) に、.pem 形式ではない有効な証明書が指定されました。	400	SetPlatformAttributes
APNs — Principal is an expired certificate	APN プリンシパル (SSL 証明書) に失効した証明書が指定されました。	400	SetPlatformAttributes
APNs — Principal is not an Apple issued certificate	APN プリンシパル (SSL 証明書) に Apple によって発行されたものではない証明書が指定されました。	400	SetPlatformAttributes
APNs — Principal is not provided	APN プリンシパル (SSL 証明書) が指定されませんでした。	400	SetPlatformAttributes

エラー	説明	HTTPS ステータスコード	API アクション
APNs — Credential is not provided	APN 認証情報 (プライベートキー) が指定されませんでした。詳細については、Amazon Simple Notification Service API リファレンスの「CreatePlatformApplication」を参照してください。	400	SetPlatformAttributes
APNs — Credential are not in a valid .pem format	APN 認証情報 (プライベートキー) が有効な .pem 形式ではありません。	400	SetPlatformAttributes
FCM — serverAPIKey is not provided	FCM 認証情報 (API キー) が指定されませんでした。詳細については、Amazon Simple Notification Service API リファレンスの「CreatePlatformApplication」を参照してください。	400	SetPlatformAttributes
FCM — serverAPIKey is a null string	FCM 認証情報 (API キー) が null です。	400	SetPlatformAttributes
ADM — clientId is not provided	クライアント ID の必須文字列が提供されていません。	400	SetPlatformAttributes

エラー	説明	HTTPS ステータスコード	API アクション
ADM — clientid is a null string	クライアント ID の必須文字列が null です。	400	SetPlatformAttributes
ADM — clientsecret is not provided	必須のクライアントシークレットが提供されていません。	400	SetPlatformAttributes
ADM — clientsecret is a null string	クライアントシークレットの必須文字列が null です。	400	SetPlatformAttributes
EventEndpointUpdated has invalid ARN format	EventEndpointUpdated が無効な ARN 形式です。	400	SetPlatformAttributes
EventEndpointDeleted has invalid ARN format	EventEndpointDeleted が無効な ARN 形式です。	400	SetPlatformAttributes
EventEndpointUpdated has invalid ARN format	EventEndpointUpdated が無効な ARN 形式です。	400	SetPlatformAttributes
EventDeliveryAttemptFailure has invalid ARN format	EventDeliveryAttemptFailure が無効な ARN 形式です。	400	SetPlatformAttributes
EventDeliveryFailure has invalid ARN format	EventDeliveryFailure が無効な ARN 形式です。	400	SetPlatformAttributes
EventEndpointCreated is not an existing Topic	EventEndpointCreated は既存のトピックではありません。	400	SetPlatformAttributes

エラー	説明	HTTPS ステータスコード	API アクション
EventEndpointDeleted is not an existing Topic	EventEndpointDeleted は既存のトピックではありません。	400	SetPlatformAttributes
EventEndpointUpdated is not an existing Topic	EventEndpointUpdated は既存のトピックではありません。	400	SetPlatformAttributes
EventDeliveryAttemptFailure is not an existing Topic	EventDeliveryAttemptFailure は既存のトピックではありません。	400	SetPlatformAttributes
EventDeliveryFailure is not an existing Topic	EventDeliveryFailure は既存のトピックではありません。	400	SetPlatformAttributes
Platform ARN is invalid	プラットフォーム ARN が無効です。	400	GetPlatformApplicationAttributes
Platform ARN is valid but does not belong to the user	プラットフォーム ARN は有効ですが、このユーザーに属していません。	403	GetPlatformApplicationAttributes
Token specified is invalid	指定されたトークンが無効です。	400	ListPlatformApplications
Platform ARN is invalid	プラットフォーム ARN が無効です。	400	ListEndpointsByPlatformApplication

エラー	説明	HTTPS ステータスコード	API アクション
Platform ARN is valid but does not belong to the user	プラットフォーム ARN は有効ですが、このユーザーに属していません。	404	ListEndpointsByPlatformApplication
Token specified is invalid	指定されたトークンが無効です。	400	ListEndpointsByPlatformApplication
Platform ARN is invalid	プラットフォーム ARN が無効です。	400	DeletePlatformApplication
Platform ARN is valid but does not belong to the user	プラットフォーム ARN は有効ですが、このユーザーに属していません。	403	DeletePlatformApplication
Platform ARN is invalid	プラットフォーム ARN が無効です。	400	CreatePlatformEndpoint
Platform ARN is valid but does not belong to the user	プラットフォーム ARN は有効ですが、このユーザーに属していません。	404	CreatePlatformEndpoint
Token is not specified	トークンが指定されていません。	400	CreatePlatformEndpoint
Token is not of correct length	トークンの長さが正しくありません。	400	CreatePlatformEndpoint

エラー	説明	HTTPS ステータスコード	API アクション
Customer User data is too large	顧客のユーザーデータは UTF-8 エンコードで 2048 バイトを超えることはできません。	400	CreatePlatformEndpoint
Endpoint ARN is invalid	エンドポイント ARN が無効です。	400	DeleteEndpoint
Endpoint ARN is valid but does not belong to the user	エンドポイント ARN は有効ですが、このユーザーに属していません。	403	DeleteEndpoint
Endpoint ARN is invalid	エンドポイント ARN が無効です。	400	SetEndpointAttributes
Endpoint ARN is valid but does not belong to the user	エンドポイント ARN は有効ですが、このユーザーに属していません。	403	SetEndpointAttributes
Token is not specified	トークンが指定されていません。	400	SetEndpointAttributes
Token is not of correct length	トークンの長さが正しくありません。	400	SetEndpointAttributes
Customer User data is too large	顧客のユーザーデータは UTF-8 エンコードで 2048 バイトを超えることはできません。	400	SetEndpointAttributes

エラー	説明	HTTPS ステータスコード	API アクション
Endpoint ARN is invalid	エンドポイント ARN が無効です。	400	`GetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user	エンドポイント ARN は有効ですが、このユーザーに属していません。	403	`GetEndpointAttributes`
Target ARN is invalid	ターゲット ARN が無効です。	400	`Publish`
Target ARN is valid but does not belong to the user	ターゲット ARN は有効ですが、このユーザーに属していません。	403	`Publish`
Message format is invalid	メッセージの形式が無効です。	400	`Publish`
Message size is larger than supported by protocol/end-service	メッセージサイズがプロトコル/エンドサービスでサポートされているより大きくなっています。	400	`Publish`

モバイルプッシュ通知サービスの Amazon SNS 有効期限 (TTL) メッセージ属性を使用する

Amazon Simple Notification Service (Amazon SNS) では、モバイルプッシュ通知サービスの有効期限 (TTL) メッセージ属性の設定がサポートされています。これは、Android への送信時に Amazon デバイスメッセージング (ADM) や Firebase クラウドメッセージング (FCM) など、これをサポートするモバイルプッシュ通知サービスの Amazon SNS メッセージ本文に TTL を設定する既存の機能に追加されます。

TTL メッセージ属性は、メッセージの有効期限メタデータを指定するために使用します。この属性を使用して、Apple Push Notification Service (APNs) や FCM などのプッシュ通知サービスによってエンドポイントにメッセージが配信される時間を指定できます。何らかの理由 (モバイルデバイスがオフになっているなど) で、指定した TTL 内にメッセージが配信されなかった場合、そのメッセージは破棄され、以降その配信は試みられません。メッセージ属性内で TTL を指定するには、、 AWS ソフトウェア開発キット (SDK) AWS Management Console、またはクエリ API を使用できます。

トピック

プッシュ通知サービスの TTL メッセージ属性

AWS SDK またはクエリ API を使用するときに設定できるプッシュ通知サービスの TTL メッセージ属性のリストを以下に示します。

プッシュ通知サービス	TTL メッセージ属性
Amazon Device Messaging (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Apple Push Notification Service (APNS)	`AWS.SNS.MOBILE.APNS.TTL`
Apple Push Notification Service Sandbox (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase クラウドメッセージング (Android への送信時は FCM)	`AWS.SNS.MOBILE.FCM.TTL`
Windows Push Notification Services (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

各プッシュ通知サービスは TTL を個別に処理します。Amazon SNS では、すべてのプッシュ通知サービスに対する TTL の抽象ビューが提供されるため、TTL の指定がより簡単になります。を使用して TTL (秒単位) を指定する場合、TTL 値を 1 回入力するだけで、Amazon SNS はメッセージの発行時に選択した各プッシュ通知サービスの TTL を計算します。 AWS Management Console

TTL は発行時間を基準にします。特定のプッシュ通知サービスにプッシュ通知メッセージを発行する前に、Amazon SNS はプッシュ通知のドウェル時間 (発行のタイムスタンプからプッシュ通知サービスへの発行直前までの時間) を計算し、残りの TTL をそのプッシュ通知サービスに渡します。TTL がドウェル時間よりも短い場合、Amazon SNS はプッシュ通知メッセージの発行を試みません。

プッシュ通知メッセージに TTL を指定する場合、TTL `0` 値は正の整数でなければなりません。ただし、の値がプッシュ通知サービスにとって特定の意味を持つ場合 (APN や FCM (Android に送信する場合) など) を除きます。TTL 値を `0` に設定した場合、プッシュ通知サービスに対して `0` に特定の意味がないと、Amazon SNS はメッセージを破棄します。APN を使用した場合に `0` に設定した TTL パラメータの詳細については、バイナリプロバイダー API ドキュメントの表 A-3 リモート通知のアイテム識別子を参照してください。

TTL を決定するための優先順位

Amazon SNS がプッシュ通知メッセージの TTL を決定するための優先順位は以下の順序に基づきます。最も小さい番号が最も高い優先順位を表しています。

1. メッセージ属性の TTL
2. メッセージ本文の TTL
3. プッシュ通知サービスのデフォルト TTL (サービスごとに異なる)
4. Amazon SNS のデフォルト TTL (4 週間)

同じメッセージに対して (メッセージ属性とメッセージ本文で) 異なる TTL 値を設定した場合、Amazon SNS はメッセージ属性の TTL に一致するようにメッセージ本文の TTL を変更します。

TTL を指定するには AWS Management Console

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで [モバイル]、[プッシュ通知] を選択します。
3. [モバイルプッシュ通知] ページの [プラットフォームアプリケーション] セクションで、アプリケーションを選択します。
4. ***MyApplication***ページの「エンドポイント」セクションで、アプリケーションエンドポイントを選択し、「Publish message」を選択します。
5. [メッセージの詳細] セクションで、TTL (プッシュ通知サービスがエンドポイントに配信される秒数) を入力します。
6. [メッセージの発行] を選択します。

モバイルアプリケーションでサポートされているリージョン

現在、モバイルアプリケーションは、次のリージョンで作成できます。

- 米国東部 (オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 欧州 (フランクフルト)
- ヨーロッパ (アイルランド)
- ヨーロッパ (ロンドン)
- ヨーロッパ (ミラノ)
- ヨーロッパ (パリ)
- ヨーロッパ (ストックホルム)
- 中東 (バーレーン)
- 中東 (アラブ首長国連邦)
- 南米 (サンパウロ)
- AWSGovCloud(米国西部)

モバイルプッシュ通知のベストプラクティス

このセクションでは、顧客エンゲージメントを高めるのに役立つ可能性があるベストプラクティスについて説明します。

エンドポイントの管理

デバイス上でのユーザーのアクション (アプリをデバイスに再インストールするなど) や、特定の iOS バージョンで実行されているデバイスに影響を与える証明書の更新によってデバイストークンが変更された場合、配信に関する問題が発生する可能性があります。Apple が推奨するベストプラクティスは、アプリを起動するたびに APN を使用して登録することです。

デバイストークンは、ユーザーがアプリを開くたびに変更されるわけではないので、べき等 `CreatePlatformEndpoint` API を使用できます。ただし、これにより、トークン自体が無効な場合や、エンドポイントが有効だが無効な場合 (たとえば、本番環境とサンドボックス環境の不一致) 、同じデバイスに対して重複が発生する可能性があります。

デバイストークン管理メカニズム (擬似コードのものなど) が使用できます。

FCM v1 デバイストークンの管理と保守については、を参照してください。Firebase クラウドメッセージング (FCM) エンドポイント管理

配信ステータスのログ記録

プッシュ通知の配信ステータスをモニタリングするには、Amazon SNS プラットフォームアプリケーションの配信ステータスログ記録を有効にすることをお勧めします。ログにはプッシュプラットフォームサービスから返されたプロバイダレスポンスコードが含まれているため、配信が失敗したときのトラブルシューティングに役立ちます。配信ステータスログ記録の有効化の詳細については、「プッシュ通知のために Amazon SNS トピック配信ログにアクセスするにはどうすればよいですか」を参照してください。

イベント通知

イベント駆動方式でエンドポイントを管理するには、イベント通知機能が使用できます。これにより、設定された Amazon SNS トピックは、エンドポイントの作成、削除、更新、および配信失敗のプラットフォームアプリケーションイベントについて、Lambda 関数などのサブスクライバーにイベントをファンアウトできます。

E メール通知

このページでは、、 AWS Management Console AWS SDK for Java、 AWS SDK for .NETまたはを使用して Amazon SNS トピックに E メールアドレスを登録する方法について説明します。

メモ

- E メール本文をカスタマイズすることはできません。E メール配信機能は、マーケティングメッセージではなく、内部システムアラートを提供することを目的としています。
- E メールエンドポイントの直接的なサブスクライブは、標準トピックのみでサポートされています。
- E メール配信のスループットは、Amazon SNS クォータに応じてスロットリングされます。

Important

Amazon SNS トピック E メールのメーリングリストの受信者全員に対して、受信者登録を解除できないようにするには、 AWS サポートからの登録解除に認証が必要な E メール受信登録を設定するを参照してください。

を使用して Amazon SNS トピックに E メールアドレスを登録するには AWS Management Console

1. Amazon SNS コンソールにサインインします。
2. 左のナビゲーションペインで、[サブスクリプション] を選択します。
3. [サブスクリプション] ページで [サブスクリプションの作成] を選択します。
4. [サブスクリプションの作成] ページで [詳細] セクションで、以下を実行します。
   a. [トピック ARN] で、トピックの Amazon リソースネーム (ARN) を選択します。
   b. [プロトコル] で [E メール] を選択します。
   c. [エンドポイント] に E メールアドレスを入力します。

d. (オプション) フィルターポリシーを設定するには、[サブスクリプションのフィルターポリシー] セクションを展開します。詳細については、「Amazon SNS サブスクリプションフィルターポリシー」を参照してください。

e. (オプション) ペイロードベースのフィルタリングを有効にするには、`Filter Policy Scope` を `MessageBody` に設定します。詳細については、「Amazon SNS サブスクリプションフィルターポリシーの範囲」を参照してください。

f. (オプション) サブスクリプションのデッドレターキューを設定するには、Redrive ポリシー (デッドレターキュー)を展開します。詳細については、「Amazon SNS デッドレターキュー (DLQ)」を参照してください。

g. [サブスクリプションの作成] を選択します。

   コンソールがサブスクリプションを作成し、サブスクリプションの [詳細] ページが開きます。

E メールアドレスがメッセージの受信を開始する前に、サブスクリプションを確認する必要があります。

サブスクリプションを確認するには

1. E メールの受信トレイを確認し、Amazon SNS からの E メールで [サブスクリプションの確認] を選択します。
2. Amazon SNS がウェブブラウザを開き、サブスクリプション ID とともにサブスクリプションの確認を表示します。

SDK を使用して Amazon SNS トピックに E メールアドレスを登録するには AWS

AWS SDK を使用するには、認証情報を使用して設定する必要があります。詳細については、「AWS SDK とツールのリファレンスガイド」の「共有設定ファイルと認証情報ファイル」を参照してください。

次のコード例は使用方法を示していますSubscribe。

.NET

AWS SDK for .NET

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックにメールアドレスを登録する。

```
/// <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;
}
```

オプションのフィルターでトピックにキューをサブスクライブします。

```
    /// <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;
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「Subscribe」を参照してください。

C++

SDK for C++

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックにメールアドレスを登録する。

```
//! 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();
}
```

モバイルアプリケーションをトピックに登録する。

```
//! 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();
}
```

Lambda 関数をトピックにサブスクライブします。

```
//! 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();
}
```

SQS キューをトピックにサブスクライブします。

```
        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;
           }
```

フィルターを使ってトピックを購読する。

```
       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;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「Subscribe」を参照してください。

CLI

AWS CLI

トピックにサブスクライブするには

次の subscribe コマンドは、指定したトピックに E メールアドレスをサブスクライブします。

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

出力:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「Subscribe」を参照してください。

Go

SDK for Go V2

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

オプションのフィルターでトピックにキューをサブスクライブします。

```
// 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
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「Subscribe」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックにメールアドレスを登録する。

```
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);
        }
    }
}
```

HTTP エンドポイントをトピックにサブスクライブします。

```
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);
        }
    }
}
```

Lambda 関数をトピックにサブスクライブします。

```
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 "";
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「Subscribe」を参照してください。

JavaScript

JavaScript (v3) 用の SDK

> **Note**
>
> にはまだまだあります。 GitHub用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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'
  // }
};
```

モバイルアプリケーションをトピックに登録する。

```
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;
};
```

Lambda 関数をトピックにサブスクライブします。

```
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;
};
```

SQS キューをトピックにサブスクライブします。

```
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;
};
```

フィルターを使ってトピックを購読する。

```
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;
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for JavaScript API リファレンスの「Subscribe」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックにメールアドレスを登録する。

```
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()
```

```
    }
}
```

Lambda 関数をトピックにサブスクライブします。

```
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}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Subscribe」を参照してください。

PHP

SDK for PHP

> **Note**
>
> にはまだまだあります。 GitHubAWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックにメールアドレスを登録する。

```
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());
}
```

HTTP エンドポイントをトピックにサブスクライブします。

```
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());
}
```

- API の詳細については、AWS SDK for PHP API リファレンスの「Subscribe」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> にはまだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックにメールアドレスを登録する。

```
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
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「Subscribe」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックにメールアドレスを登録する。

```
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
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for Ruby API リファレンスの「Subscribe」を参照してください。

Rust

SDK for Rust

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックにメールアドレスを登録する。

```
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(())
}
```

- API の詳細については、AWS DK for Rust API リファレンスの「Subscribe」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> まだまだあります GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックにメールアドレスを登録する。

```
   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.
```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの「Subscribe」(サブスクライブ) を参照してください。

SDK を使用した Amazon SNS のコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で Amazon SNS を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

クロスサービスの例は、複数の AWS のサービスで動作するサンプルアプリケーションです。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

開始方法

Hello Amazon SNS

以下のコード例は、Amazon SNS の使用を開始する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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();
        }
    }
}
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

C MakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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;
}
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。
  AWS SDK for C++

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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)
  }
 }
}
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for Go

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

SNS クライアントを初期化し、アカウントのトピックを一覧表示します。

```
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"));
};
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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}")
            }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスListTopicsの「」を参照してください。

コードの例

- SDK を使用した Amazon SNS のアクション AWS SDKs
  - AWS SDK または CLI CheckIfPhoneNumberIsOptedOutで を使用する
  - AWS SDK または CLI ConfirmSubscriptionで を使用する
  - AWS SDK または CLI CreateTopicで を使用する
  - AWS SDK または CLI DeleteTopicで を使用する
  - AWS SDK または CLI GetSMSAttributesで を使用する
  - AWS SDK または CLI GetTopicAttributesで を使用する
  - AWS SDK または CLI ListPhoneNumbersOptedOutで を使用する
  - AWS SDK または CLI ListSubscriptionsで を使用する

SDK を使用した Amazon SNS のアクション AWS SDKs

次のコード例は、 AWS SDKs を使用して個々の Amazon SNS アクションを実行する方法を示しています。これらは Amazon SNS API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。各例には へのリンクが含まれており GitHub、コードの設定と実行の手順を確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、「Amazon Simple Notification Service (Amazon SNS) API リファレンス」を参照してください。

例

- AWS SDK または CLI CheckIfPhoneNumberIsOptedOutで を使用する
- AWS SDK または CLI ConfirmSubscriptionで を使用する
- AWS SDK または CLI CreateTopicで を使用する
- AWS SDK または CLI DeleteTopicで を使用する
- AWS SDK または CLI GetSMSAttributesで を使用する
- AWS SDK または CLI GetTopicAttributesで を使用する
- AWS SDK または CLI ListPhoneNumbersOptedOutで を使用する
- AWS SDK または CLI ListSubscriptionsで を使用する
- AWS SDK または CLI ListTopicsで を使用する
- AWS SDK または CLI Publishで を使用する
- AWS SDK または CLI SetSMSAttributesで を使用する
- AWS SDK または CLI SetSubscriptionAttributesで を使用する
- AWS SDK または CLI SetSubscriptionAttributesRedrivePolicyで を使用する
- AWS SDK または CLI SetTopicAttributesで を使用する
- AWS SDK または CLI Subscribeで を使用する
- AWS SDK または CLI TagResourceで を使用する
- AWS SDK または CLI Unsubscribeで を使用する

AWS SDK または CLI **CheckIfPhoneNumberIsOptedOut**で を使用する

以下のコード例は、CheckIfPhoneNumberIsOptedOut の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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}");
            }
        }
    }
```

- API の詳細については、「 API リファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

電話番号での SMS メッセージのオプトアウトを確認するには

次のcheck-if-phone-number-is-opted-out例では、指定された電話番号が現在の AWS アカウントからの SMS メッセージの受信をオプトアウトされているかどうかを確認します。

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

出力:

```
{
    "isOptedOut": false
}
```

- API の詳細については、「 コマンドリファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS SDK for JavaScript

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスCheckIfPhoneNumberIsOptedOut」の「」を参照してください。 AWS SDK for PHP

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ConfirmSubscription**で を使用する

以下のコード例は、ConfirmSubscription の使用方法を示しています。

CLI

AWS CLI

サブスクリプションを確認するには

次の confirm-subscription コマンドは、my-topic という名前の SNS トピックをサブスクライブしたときに開始された確認プロセスを完了します。--token パラメータ

は、subscribe の呼び出しで指定した通知エンドポイントに送信される確認メッセージから取得されます。

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --token
 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7ce5d0c
```

出力:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- API の詳細については、「 コマンドリファレンスConfirmSubscription」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスConfirmSubscription」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスConfirmSubscription」の「」を参照してください。 AWS SDK for JavaScript

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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());
}
```

- API の詳細については、「 API リファレンスConfirmSubscription」の「」を参照してください。 AWS SDK for PHP

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `CreateTopic`で を使用する

以下のコード例は、`CreateTopic` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- FIFO トピックを作成して発行する
- メッセージをキューに発行する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックを作成して、個別の名前を付けます。

```
    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;
        }
    }
```

名前と特定の FIFO および重複除外属性を使用して新しいトピックを作成します。

```
    /// <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;
    }
```

- API の詳細については、「 API リファレンスCreateTopic」の「」を参照してください。AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「 API リファレンスCreateTopic」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

SNS トピックを作成するには

次の create-topic の例では、my-topic という名前の SNS トピックを作成します。

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

出力:

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

詳細については、AWS 「 コマンドラインインターフェイスユーザーガイド」のAmazon SQS および Amazon SNS AWS でのコマンドラインインターフェイスの使用」を参照してください。

- API の詳細については、「 コマンドリファレンスCreateTopic」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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
}
```

- API の詳細については、「 API リファレンスCreateTopic」の「」を参照してください。 AWS SDK for Go

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 "";
    }
}
```

- API の詳細については、「 API リファレンスCreateTopic」の「」を参照してください。AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスCreateTopic」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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()
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスCreateTopicの「」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスCreateTopic」の「」を参照してください。 AWS SDK for PHP

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 CreateTopic AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスCreateTopic」の「」を参照してください。 AWS SDK for Ruby

Rust

SDK for Rust

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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(())
}
```

- API の詳細については、 CreateTopic AWS SDK for Rust API リファレンスの「」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、CreateTopicAWS 「 SDK for SAP ABAP API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteTopic**で を使用する

以下のコード例は、DeleteTopic の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- メッセージをキューに発行する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピック ARN でトピックを削除します。

```
/// <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;
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

SNS トピックを削除するには

次の `delete-topic` の例では、指定した SNS トピックを削除します。

```
aws sns delete-topic \
```

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

このコマンドでは何も出力されません。

- API の詳細については、「 コマンドリファレンスDeleteTopic」の「」を参照してください。 AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for Go

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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
  //   }
  // }
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスDeleteTopicの「」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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());
}
```

- API の詳細については、「 API リファレンスDeleteTopic」の「」を参照してください。 AWS SDK for PHP

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 DeleteTopic AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、DeleteTopicAWS 「 SDK for SAP ABAP API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetSMSAttributes**で を使用する

以下のコード例は、GetSMSAttributes の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「GetSMSAttributes」を参照してください。

CLI

AWS CLI

デフォルトの SMS メッセージ属性を一覧表示するには

次の `get-sms-attributes` の例では、SMS メッセージを送信するためのデフォルト属性を一覧表示しています。

```
aws sns get-sms-attributes
```

出力:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetSMSAttributes」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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");
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「GetSMSAttributes」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、AWS SDK for JavaScript API リファレンスの「GetSMSAttributes」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API リファレンスの「GetSMSAttributes」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetTopicAttributes**で を使用する

以下のコード例は、GetTopicAttributes の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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");
            }
        }
    }
```

- API の詳細については、「 API リファレンスGetTopicAttributes」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「 API リファレンスGetTopicAttributes」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

トピックの属性を取得するには

次の get-topic-attributes の例では、指定したトピックの属性を表示します。

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

出力:

```
{
    "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"
    }
}
```

- API の詳細については、「 コマンドリファレンスGetTopicAttributes」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスGetTopicAttributes」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスGetTopicAttributes」の「」を参照してください。 AWS SDK for JavaScript

SDK for JavaScript (v2)

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
// 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);
  });
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスGetTopicAttributes」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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}")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスGetTopicAttributesの「」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
$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());
}
```

- API の詳細については、「 API リファレンスGetTopicAttributes」の「」を参照してください。 AWS SDK for PHP

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、GetTopicAttributesAWS 「 SDK for SAP ABAP API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `ListPhoneNumbersOptedOut` で を使用する

以下のコード例は、`ListPhoneNumbersOptedOut` の使用方法を示しています。

CLI

AWS CLI

SMS メッセージのオプトアウトを一覧表示するには

次の `list-phone-numbers-opted-out` の例では、SMS メッセージの受信をオプトアウトした電話番号を一覧表示しています。

```
aws sns list-phone-numbers-opted-out
```

出力:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- API の詳細については、「 コマンドリファレンスListPhoneNumbersOptedOut」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスListPhoneNumbersOptedOut」の「」を参照してください。 AWS SDK for Java 2.x

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスListPhoneNumbersOptedOut」の「」を参照してください。 AWS SDK for PHP

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListSubscriptions**で を使用する

以下のコード例は、ListSubscriptions の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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();
            }
        }
    }
```

- API の詳細については、「 API リファレンスListSubscriptions」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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;
}
```

- API の詳細については、「 API リファレンスListSubscriptions」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

SNS サブスクリプションを一覧表示するには

次のlist-subscriptions例では、 AWS アカウントの SNS サブスクリプションのリストを表示します。

```
aws sns list-subscriptions
```

出力:

```
{
    "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"
        }
    ]
}
```

- API の詳細については、「 コマンドリファレンスListSubscriptions」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスListSubscriptions」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスListSubscriptions」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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}")
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスListSubscriptionsの「」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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());
}
```

- API の詳細については、「 API リファレンスListSubscriptions」の「」を参照してください。 AWS SDK for PHP

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 ListSubscriptions AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスListSubscriptions」の「」を参照してください。 AWS SDK for Ruby

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、ListSubscriptionsAWS 「 SDK for SAP ABAP API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListTopics**で を使用する

以下のコード例は、ListTopics の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   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}");
            }
        }
    }
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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;
}
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

SNS トピックを一覧表示するには

次のlist-topics例では、アカウント内のすべての SNS トピックを一覧表示します AWS 。

```
aws sns list-topics
```

出力:

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- API の詳細については、「 コマンドリファレンスListTopics」の「」を参照してください。AWS CLI

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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)
  }
 }
}
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for Go

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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}")
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスListTopicsの「」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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());
}
```

- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。
  AWS SDK for PHP

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 ListTopics AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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__
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスListTopics」の「」を参照してください。 AWS SDK for Ruby

Rust

SDK for Rust

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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(())
}
```

- API の詳細については、 ListTopics AWS SDK for Rust API リファレンスの「」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、ListTopicsAWS 「 SDK for SAP ABAP API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **Publish**で を使用する

以下のコード例は、Publish の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- FIFO トピックを作成して発行する
- SMS テキストメッセージを発行する
- メッセージをキューに発行する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックへのメッセージの発行

```
    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}");
        }
    }
```

グループ、重複、属性のオプションを指定してメッセージをトピックに発行します。

```
    /// <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);
        }
    }
```

ユーザーの選択を発行アクションに適用します。

```
    /// <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;
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「Publish」を参照してください。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

属性を含むメッセージを発行します。

```
        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;
        }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「Publish」を参照してください。

CLI

AWS CLI

例 1: トピックにメッセージを発行するには

次の publish の例では、指定した Amazon SNS トピックに指定した通知を公開します。メッセージはテキストファイルから取得されたもので、改行を含めることができます。

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

message.txt の内容:

```
Hello World
Second Line
```

出力:

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

例 2: 電話番号に SMS メッセージを公開するには

次の publish の例では、Hello world! メッセージを電話番号 +1-555-555-0100 に公開します。

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

出力:

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

- API の詳細については、「AWS CLI コマンドリファレンス」の「Publish」を参照してください。

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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
}
```

- API の詳細については、AWS SDK for Go API リファレンスの「発行」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「Publish」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

グループ、重複、属性のオプションを指定してメッセージをトピックに発行します。

```
  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();
    }
  }
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for JavaScript API リファレンスの「発行」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Publish」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API リファレンスの「発行」を参照してください。

PowerShell

のツール PowerShell

例 1: この例は、インラインで MessageAttribute 宣言された 1 つのメッセージを発行する例を示しています。

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

例 2: この例は、事前に複数の が MessageAttributes 宣言されたメッセージを発行することを示しています。

```
$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
```

- API の詳細については、「 コマンドレットリファレンス」の「パブリッシュ」を参照してください。 AWS Tools for PowerShell

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

サブスクリプションが属性に基づいてフィルター処理できるように、属性を含むメッセージを発行します。

```
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
```

受信者のプロトコルに基づいて異なる形式のメッセージを発行します。

```
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
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「Publish」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for Ruby API リファレンスの「発行」を参照してください。

Rust

SDK for Rust

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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(())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「発行」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの「Publish」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **SetSMSAttributes**で を使用する

以下のコード例は、SetSMSAttributes の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon SNS を使用して defaultsMsType 属性を設定する方法。

```
//! 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();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「SetSMSAttributes」を参照してください。

CLI

AWS CLI

SMS メッセージ属性を設定するには

次の `set-sms-attributes` の例では、SMS メッセージのデフォルトの送信者 ID を `MyName` に設定します。

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「SetSMSAttributes」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
```

```
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「SetSMSAttributes」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、AWS SDK for JavaScript API リファレンスの「SetSMSAttributes」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
$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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API リファレンスの「SetSMSAttributes」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **SetSubscriptionAttributes**で を使用する

以下のコード例は、SetSubscriptionAttributes の使用方法を示しています。

CLI

AWS CLI

サブスクリプション属性を設定するには

次の set-subscription-attributes の例では、RawMessageDelivery 属性を SQS サブスクリプションに設定します。

```
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
```

このコマンドでは何も出力されません。

次の set-subscription-attributes の例では、FilterPolicy 属性を SQS サブスクリプションに設定します。

```
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\"] }"
```

このコマンドでは何も出力されません。

次の set-subscription-attributes の例では、FilterPolicy 属性を SQS サブスクリプションから削除します。

```
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 "{}"
```

このコマンドでは何も出力されません。

- API の詳細については、「 コマンドリファレンスSetSubscriptionAttributes」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスSetSubscriptionAttributes」の「」を参照してください。 AWS SDK for Java 2.x

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 SetSubscriptionAttributes AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **SetSubscriptionAttributesRedrivePolicy**で を使用する

次の例は、SetSubscriptionAttributesRedrivePolicy を使用する方法を説明しています。

Java

SDK for Java 1.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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);
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **SetTopicAttributes**で を使用する

以下のコード例は、SetTopicAttributes の使用方法を示しています。

CLI

AWS CLI

トピックの属性を設定するには

次の set-topic-attributes の例では、指定したトピックの DisplayName 属性を設定します。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

このコマンドでは何も出力されません。

- API の詳細については、「 コマンドリファレンスSetTopicAttributes」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスSetTopicAttributes」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、「 API リファレンスSetTopicAttributes」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスSetTopicAttributesの「」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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());
}
```

- API の詳細については、「 API リファレンスSetTopicAttributes」の「」を参照してください。 AWS SDK for PHP

Ruby

SDK for Ruby

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
# 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
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスSetTopicAttributes」の「」を参照してください。 AWS SDK for Ruby

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.
```

- API の詳細については、SetTopicAttributesAWS 「 SDK for SAP ABAP API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **Subscribe**で を使用する

以下のコード例は、Subscribe の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- FIFO トピックを作成して発行する
- メッセージをキューに発行する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
/// <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;
}
```

オプションのフィルターでトピックにキューをサブスクライブします。

```
    /// <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;
    }
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「Subscribe」を参照してください。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
//! 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();
}
```

モバイルアプリケーションをトピックにサブスクライブします。

```
//! 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();
}
```

Lambda 関数をトピックにサブスクライブします。

```
//! 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();
}
```

SQS キューをトピックにサブスクライブします。

```
        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;
           }
```

フィルターを使用してトピックをサブスクライブします。

```
       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;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「Subscribe」を参照してください。

CLI

AWS CLI

トピックにサブスクライブするには

次の subscribe コマンドは、指定したトピックに E メールアドレスをサブスクライブします。

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

出力:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の「Subscribe」を参照してください。

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

オプションのフィルターでトピックにキューをサブスクライブします。

```
// 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
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「Subscribe」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
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);
        }
    }
}
```

HTTP エンドポイントをトピックにサブスクライブします。

```
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);
        }
    }
}
```

Lambda 関数をトピックにサブスクライブします。

```
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 "";
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「Subscribe」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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'
  // }
};
```

モバイルアプリケーションをトピックにサブスクライブします。

```
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;
};
```

Lambda 関数をトピックにサブスクライブします。

```
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;
};
```

SQS キューをトピックにサブスクライブします。

```
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;
};
```

フィルターを使用してトピックをサブスクライブします。

```
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;
};
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for JavaScript API リファレンスの「Subscribe」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
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()
```

```
    }
}
```

Lambda 関数をトピックにサブスクライブします。

```
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}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Subscribe」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
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());
}
```

HTTP エンドポイントをトピックにサブスクライブします。

```
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());
}
```

- API の詳細については、AWS SDK for PHP API リファレンスの「Subscribe」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
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
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「Subscribe」を参照してください。

Ruby

SDK for Ruby

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
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
```

- 詳細については、「AWS SDK for Ruby デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for Ruby API リファレンスの「Subscribe」を参照してください。

Rust

SDK for Rust

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
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(())
}
```

- API の詳細については、AWS DK for Rust API リファレンスの「Subscribe」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
   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.
```

- API の詳細については、AWS SDK for SAP ABAP API リファレンスの「Subscribe」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **TagResource**で を使用する

以下のコード例は、TagResource の使用方法を示しています。

CLI

AWS CLI

トピックにタグを追加するには

次の tag-resource の例では、指定した Amazon SNS トピックにメタデータタグを追加します。

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

このコマンドでは何も出力されません。

- API の詳細については、「 コマンドリファレンスTagResource」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「 API リファレンスTagResource」の「」を参照してください。 AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスTagResourceの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **Unsubscribe**で を使用する

以下のコード例は、Unsubscribe の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- メッセージをキューに発行する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

サブスクリプション ARN でトピックからサブスクライブを解除します。

```
/// <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;
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「Unsubscribe」を参照してください。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「Unsubscribe」を参照してください。

CLI

AWS CLI

トピックからサブスクライブを解除するには

次の unsubscribe の例では、指定したサブスクリプションをトピックから削除します。

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

このコマンドでは何も出力されません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「Unsubscribe」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「Unsubscribe」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

別のモジュールでクライアントを作成し、エクスポートします。

```
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({});
```

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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;
};
```

- 詳細については、AWS SDK for JavaScript デベロッパーガイドを参照してください。
- API の詳細については、AWS SDK for JavaScript API リファレンスの「Unsubscribe」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Unsubscribe」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API リファレンスの「Unsubscribe」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「Unsubscribe」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「Unsubscribe」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した Amazon SNS のシナリオ AWS SDKs

次のコード例は、 AWS SDKs を使用して Amazon SNS に一般的なシナリオを実装する方法を示しています。これらのシナリオは、Amazon SNS 内で複数の関数を呼び出して特定のタスクを実行する方法を示します。各シナリオには GitHub、コードの設定と実行の手順を示す へのリンクが含まれています。

例

- AWS SDK を使用して Amazon SNS プッシュ通知用のプラットフォームエンドポイントを作成する
- AWS SDK を使用して FIFO Amazon SNS トピックを作成して発行する
- AWS SDK を使用して Amazon SNS トピックに SMS メッセージを発行する
- AWS SDK を使用して Amazon S3 で大きなメッセージを Amazon SNS Amazon S3に発行する
- AWS SDK を使用して Amazon SNS SMS テキストメッセージを発行する
- AWS SDK を使用して Amazon SNS メッセージを Amazon SQS キューに発行する

AWS SDK を使用して Amazon SNS プッシュ通知用のプラットフォームエンドポイントを作成する

次のコード例は、Amazon SNS プッシュ通知のプラットフォームエンドポイントを作成する方法を示しています。

CLI

AWS CLI

プラットフォームアプリケーションのエンドポイントを作成するには

次の `create-platform-endpoint` の例では、指定したトークンを使用して、指定したプラットフォームアプリケーションのエンドポイントを作成します。

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

出力:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して FIFO Amazon SNS トピックを作成して発行する

次のコード例は、FIFO Amazon SNS トピックを作成、発行する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

この例では

- 1 つの Amazon SNS FIFO トピック、2 つの Amazon SQS FIFO キュー、および 1 つの標準キューを作成します。
- キューをトピックにサブスクライブし、メッセージをトピックに発行します。

テストでは、各キューへのメッセージの受信を検証します。完全な例では、アクセスポリシーの追加と、最後にリソースの削除も示しています。

```
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);
        }
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - CreateTopic

- Publish
- Subscribe

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

FIFO トピックを作成し、そのトピックに Amazon SQS FIFO キューと標準キューをサブスクライブして、メッセージを Amazon SNS トピックに発行します。

```
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
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - CreateTopic
  - Publish
  - Subscribe

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

FIFO トピックを作成し、そのトピックに Amazon SQS FIFO キューをサブスクライブして、Amazon SNS トピックにメッセージを発行します。

```
    " 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.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の以下のトピックを参照してください。
  - CreateTopic
  - Publish
  - Subscribe

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon SNS トピックに SMS メッセージを発行する

次のコードの例は以下の方法を示しています。

- Amazon SNS トピックを作成します。
- 携帯電話番号をトピックにサブスクライブする。
- SMS メッセージをトピックに発行して、登録されているすべての電話番号がメッセージを一度に受信できるようにします。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

トピックを作成し、その ARN を返します。

```
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 "";
    }
}
```

トピックへのエンドポイントのサブスクライブ。

```
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);
        }
    }
}
```

送信者の ID、上限価格、タイプなど、メッセージ上の属性を設定します。メッセージ属性はオプションです。

```
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);
```

```
        }
    }
}
```

トピックへのメッセージの発行 メッセージは、すべての受信者に送信されます。

```
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);
        }
    }
}
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前のSDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 で大きなメッセージを Amazon SNS Amazon S3に発行する

次のコード例は、Amazon S3 を使用してメッセージペイロードを保存する大きなメッセージをAmazon SNS に発行する方法を示しています。

Java

SDK for Java 1.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

容量の大きなメッセージを発行するには、Amazon SNS Extended Client Library for Java を使用します。送信するメッセージは、実際のメッセージコンテンツを含む Amazon S3 オブジェクトをリファレンスします。

```
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());
        }
}
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon SNS SMS テキストメッセージを発行する

次のコード例は、Amazon SNS を使用して SMS メッセージを発行する方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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}");
            }
        }
    }
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「発行」を参照してください。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * 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();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「発行」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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);
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「Publish」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「Publish」を参照してください。

PHP

SDK for PHP

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
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());
}
```

- 詳細については、「AWS SDK for PHP デベロッパーガイド」を参照してください。
- API の詳細については、AWS SDK for PHP API リファレンスの「発行」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「Publish」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon SNS メッセージを Amazon SQS キューに発行する

次のコード例は、以下を実行する方法を示しています。

- トピック (FIFO または非 FIFO) を作成します。
- フィルターを適用するオプションを使用して、複数のキューをトピックにサブスクライブします。
- メッセージをトピックに発行します。
- キューをポーリングして受信メッセージを確認します。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
/// <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;
    }
}
```

Amazon SQS オペレーションをラップするクラスを作成します。

```
/// <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;
```

```
    }
}
```

Amazon SNS オペレーションをラップするクラスを作成します。

```
/// <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;
    }
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
        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();
}
```

- API の詳細については、『AWS SDK for C++ API リファレンス』の以下のトピックを参照してください。

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Publish
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

Go

SDK for Go V2

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
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))
}
```

この例で使用されている Amazon SNS アクションをラップする構造体を定義します。

```
// 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
}
```

この例で使用されている Amazon SQS アクションをラップする構造体を定義します。

```
// 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
}
```

- API の詳細については、『AWS SDK for Go API リファレンス』の以下のトピックを参照してください。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

これがこのワークフローのエントリーポイントです。

```
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();
};
```

上記のコードにより必要な依存関係が提供され、ワークフローが開始されます。次のセクションには、この例の大部分が含まれています。

```
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();
    }
  }
}
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs を使用した Amazon SNS のサーバーレスの例

次のコード例は、 SDK で Amazon SNS を使用する方法を示しています。 AWS SDKs

例

- Amazon SNS トリガーから Lambda 関数を呼び出す

Amazon SNS トリガーから Lambda 関数を呼び出す

次のコード例は、SNS トピックからメッセージを受信することによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数はイベントパラメータからメッセージを取得し、各メッセージの内容を記録します。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行する方法を確認してください。

.NET を使用した Lambda での SNS イベントの消費。

```
// 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 for Go V2

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Go を使用して Lambda で SNS イベントを消費します。

```
// 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 for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での SNS イベントの消費。

```
// 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 for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

を使用して Lambda で SNS イベントを消費する 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;
  }
}
```

を使用して Lambda で SNS イベントを消費する 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**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

PHP を使用して Lambda で SNS イベントを消費します。

```
// 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 for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Python を使用して Lambda で SNS イベントを消費します。

```
# 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 for Ruby

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Ruby を使用した Lambda での SNS イベントの消費。

```
# 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 for Rust

> **Note**
>
> については、「」を参照してください GitHub。サーバーレスサンプルリポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Rust を使用して Lambda で SNS イベントを消費します。

```
// 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
}
```

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs を使用した Amazon SNS のクロスサービスの例

次のサンプルアプリケーションでは、 AWS SDKsを使用して Amazon SNS を他の と組み合わせます AWS のサービス。各例には GitHub、アプリケーションのセットアップと実行の手順を示す へのリンクが含まれています。

例

- DynamoDB テーブルにデータを送信するアプリケーションを構築する
- メッセージを翻訳する公開およびサブスクリプションアプリケーションを構築する
- ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成
- Amazon Textract エクスプローラーアプリケーションを作成する
- AWS SDK を使用して Amazon Rekognition でビデオ内のユーザーとオブジェクトを検出する
- AWS SDK を使用して Amazon SNS メッセージを Amazon SQS キューに発行する
- API Gateway を使用して Lambda 関数を呼び出す
- スケジュールされたイベントを使用した Lambda 関数の呼び出し

DynamoDB テーブルにデータを送信するアプリケーションを構築する

次のコード例は、Amazon DynamoDB テーブルにデータを送信し、ユーザーがテーブルを更新したときに通知するアプリケーションを構築する方法を示しています。

Java

SDK for Java 2.x

Amazon DynamoDB Java API を使用してデータを送信し、Amazon Simple Notification Service Java API を使用してテキストメッセージを送信する動的ウェブアプリケーションを作成する方法について説明します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- DynamoDB
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

この例では、ユーザーが Amazon DynamoDB テーブルにデータを送信し、Amazon Simple Notification Service (Amazon SNS) を使用して管理者にテキストメッセージを送信できるようにするアプリを構築する方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例は、AWS SDK for JavaScript v3 デベロッパーガイドでも使用できます。

この例で使用されているサービス

- DynamoDB
- Amazon SNS

Kotlin

SDK for Kotlin

Amazon DynamoDB Kotlin API を使用してデータを送信し、Amazon SNS Kotlin API を使用してテキストメッセージを送信するネイティブ Android アプリケーションを作成する方法を示しています。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- DynamoDB
- Amazon SNS

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

メッセージを翻訳する公開およびサブスクリプションアプリケーションを構築する

次のコード例は、サブスクリプションと発行機能を持ち、メッセージを翻訳するアプリケーションを作成する方法を示しています。

.NET

AWS SDK for .NET

Amazon Simple Notification Service .NET API を使用して、サブスクリプションおよびパブリッシュ機能を持つウェブアプリケーションを作成する方法を説明します。さらに、このサンプルアプリケーションではメッセージを翻訳します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon SNS
- Amazon Translate

Java

SDK for Java 2.x

Amazon Simple Notification Service Java API を使用して、サブスクリプションおよびパブリッシュ機能を持つウェブアプリケーションを作成する方法を説明します。さらに、このサンプルアプリケーションではメッセージを翻訳します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

完全なソースコードと、Java 非同期 API を使用する例をセットアップして実行する方法については、 の完全な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon SNS
- Amazon Translate

Kotlin

SDK for Kotlin

Amazon SNS Kotlin API を使用して、サブスクリプションおよび発行機能を持つアプリケーションを作成する方法を示しています。さらに、このサンプルアプリケーションではメッセージを翻訳します。

完全なソースコードとウェブアプリケーションの作成方法については、「」の詳細な例を参照してくださいGitHub。

完全なソースコードとネイティブ Android アプリの作成方法については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon SNS
- Amazon Translate

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成

以下のコード例は、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションの作成方法を示しています。

.NET

AWS SDK for .NET

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

C++

SDK for C++

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK for Java 2.x

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK for Kotlin

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK for PHP

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK for Rust

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してください GitHub。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Textract エクスプローラーアプリケーションを作成する

次のコード例は、インタラクティブアプリケーションで Amazon Textract の出力を調べる方法を示しています。

JavaScript

SDK for JavaScript (v3)

を使用して AWS SDK for JavaScript 、Amazon Textract を使用してドキュメントイメージからデータを抽出し、インタラクティブなウェブページに表示する React アプリケーションを構築する方法を示します。この例はウェブブラウザで実行され、認証情報に認証された Amazon Cognito ID が必要です。Amazon Simple Storage Service (Amazon S3) をストレージに使用し、通知のために、Amazon Simple Notification Service (Amazon SNS) トピックにサブスクライブした Amazon Simple Queue Service (Amazon SQS) キューをポーリングします。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Cognito ID
- Amazon S3
- Amazon SNS

- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

Amazon Textract AWS SDK for Python (Boto3) で を使用して、ドキュメントイメージ内のテキスト、フォーム、およびテーブル要素を検出する方法を示します。入力イメージと Amazon Textract 出力は、検出された要素を探索できる Tkinter アプリケーションに表示されます。

- Amazon Textract にドキュメントイメージを送信し、検出された要素の出力を調べます。
- Amazon Textract に直接イメージを送信するか、Amazon Simple Storage Service (Amazon S3) バケットを通じてイメージを送信します。
- 非同期 API を使用して、ジョブの完了時に Amazon Simple Notification Service (Amazon SNS) トピックに通知を発行するジョブを開始します。
- Amazon Simple Queue Service (Amazon SQS) キューにジョブ完了メッセージについてポーリングし、結果を表示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon Rekognition でビデオ内のユーザーとオブジェクトを検出する

次のコード例は、Amazon Rekognition で動画内の人やオブジェクトを検出する方法を示します。

Python

SDK for Python (Boto3)

Amazon Rekognition を使用して、非同期検出ジョブを開始して、動画内の顔、オブジェクト、人を検出します。この例では、ジョブが完了し、Amazon Simple Queue Service (Amazon SQS) キューをトピックにサブスクライブしたときに、Amazon Simple Notification Service (Amazon SNS) トピックに通知するように Amazon Rekognition を設定します。キューがジョブに関するメッセージを受信すると、ジョブが取得され、結果が出力されます。

この例は、 で表示するのが最適です GitHub。完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon SNS メッセージを Amazon SQS キューに発行する

次のコード例は、以下を実行する方法を示しています。

- トピック (FIFO または非 FIFO) を作成します。
- フィルターを適用するオプションを使用して、複数のキューをトピックにサブスクライブします。
- メッセージをトピックに発行します。
- キューをポーリングして受信メッセージを確認します。

Java

SDK for Java 2.x

Amazon Simple Notification Service (Amazon SNS) と Amazon Simple Queue Service (Amazon SQS) でのトピックとキューを使用したメッセージングを示します。

Amazon SNS および Amazon SQS のトピックとキューを使用したメッセージングを示す完全なソースコードと手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon SNS
- Amazon SQS

Kotlin

SDK for Kotlin

Amazon Simple Notification Service (Amazon SNS) と Amazon Simple Queue Service (Amazon SQS) でのトピックとキューを使用したメッセージングを示します。

Amazon SNS および Amazon SQS のトピックとキューを使用したメッセージングを示す完全なソースコードと手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon SNS
- Amazon SQS

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

API Gateway を使用して Lambda 関数を呼び出す

次のコード例は、Amazon API Gateway によって呼び出される AWS Lambda 関数を作成する方法を示しています。

Java

SDK for Java 2.x

Lambda Java ランタイム API を使用して AWS Lambda 関数を作成する方法を示します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、Amazon API Gateway によって呼び出される Lambda 関数を作成する方法を示します。この関数は、Amazon DynamoDB テーブルをスキャンして、Amazon Simple Notification

Service (Amazon SNS) を使用して、従業員に年間の記念日を祝福するテキストメッセージを送信します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

Lambda JavaScript ランタイム API を使用して AWS Lambda 関数を作成する方法を示します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、Amazon API Gateway によって呼び出される Lambda 関数を作成する方法を示します。この関数は、Amazon DynamoDB テーブルをスキャンして、Amazon Simple Notification Service (Amazon SNS) を使用して、従業員に年間の記念日を祝福するテキストメッセージを送信します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例は、AWS SDK for JavaScript v3 デベロッパーガイドでも使用できます。

この例で使用されているサービス

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

スケジュールされたイベントを使用した Lambda 関数の呼び出し

次のコード例は、Amazon EventBridge スケジュールされたイベントによって呼び出される AWS Lambda 関数を作成する方法を示しています。

Java

SDK for Java 2.x

AWS Lambda 関数を呼び出す Amazon EventBridge スケジュールイベントを作成する方法を示します。cron 式 EventBridge を使用して Lambda 関数が呼び出されるタイミングをスケジュールするように を設定します。この例では、Lambda Java ランタイム API を使用して Lambda 関数を作成します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、年間の記念日に従業員を祝福するモバイルテキストメッセージを従業員に送信するアプリを作成する方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

AWS Lambda 関数を呼び出す Amazon EventBridge スケジュールイベントを作成する方法を示します。cron 式 EventBridge を使用して Lambda 関数が呼び出されるタイミングをスケジュールするように を設定します。この例では、Lambda JavaScript ランタイム API を使用して Lambda 関数を作成します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、年間の記念日に従業員を祝福するモバイルテキストメッセージを従業員に送信するアプリを作成する方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例は、AWS SDK for JavaScript v3 デベロッパーガイドでも使用できます。

この例で使用されているサービス

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での Amazon SNS の使用。このトピックには、使用開始方法に関する情報と、以前のSDK バージョンの詳細も含まれています。

Amazon SNS セキュリティ

このセクションでは、Amazon SNS のセキュリティ、認証、アクセスコントロール、Amazon SNS アクセスポリシー言語について説明します。

トピック

データ保護

AWS の責任共有モデルは、Amazon Simple Notification Service でのデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護する責任を担います。ユーザーには、このインフラストラクチャでホストされているコンテンツに対する制御を維持する責任があります。このコンテンツには、使用する AWS のサービスに対するセキュリティの設定と管理タスクが含まれます。データプライバシーの詳細については、 データプライバシーのよくある質問 を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データを保護するため、AWS アカウント の認証情報を保護し、AWS Identity and Access Management (IAM) を使用して個々のユーザーアカウントをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 以降が推奨されています。
- AWS CloudTrail で API とユーザーアクティビティログをセットアップします。
- AWS 暗号化ソリューションを AWS のサービス内のすべてのデフォルトのセキュリティ管理と一緒に使用します。

- Amazon Macie などのアドバンストマネージドセキュリティサービスを使用します。これは、Amazon S3 に保存されている個人データの検出と保護を支援します。
- コマンドラインインターフェースまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。使用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。
- メッセージデータ保護
  - メッセージデータ保護は Amazon SNS の新しい主要機能です
  - MDP を使用してメッセージをスキャンして機密情報または重要情報がないか調べる
  - トピック内を流れるすべてのコンテンツにメッセージ監査を提供する
  - トピックに公開されたメッセージおよびトピックによって配信されるメッセージへのコンテンツアクセス制御を提供する

> **Important**
>
> お客様の E メールアドレスなどの機密情報やセンシティブ情報は、タグや [Name] (名前) フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK で Amazon SNS または他の Amazon Web Services 使用する場合も同様です。タグまたは名前に使用する自由記入欄に入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへ URL を供給する場合は、そのサーバーへのリクエストを検証するために、認証情報を URL に含めないことを強くお勧めします。

次のセクションで、Amazon SNS のデータ保護についてさらに詳しく学習できます。

トピック

データ暗号化

データ保護には、転送時 (Amazon SNS との間でデータを送受信するとき) のデータを保護するものと、保管時 (Amazon SNS データセンター内のディスクに格納されているとき) のデータを保護

するものがあります。Secure Sockets Layer (SSL)またはクライアント側の暗号化を使用して、転送時のデータを保護できます。デフォルトでは、Amazon SNS はディスク暗号化を使用してメッセージとファイルを保存します。Amazon SNS にメッセージをデータセンターの暗号化されたファイルシステムに保存する前に暗号化するようリクエストすることで、保管中のデータを保護できます。Amazon SNS では、データ暗号化を最適化するために SSE を使用することをお勧めします。

トピック

- 保管中の暗号化
- キーの管理
- Amazon SNS トピックのサーバー側の暗号化 (SSE) を有効にする
- 暗号化された Amazon SQS キューをサブスクライブして Amazon SNS トピックのサーバー側の暗号化 (SSE) を有効にする

保管中の暗号化

サーバー側の暗号化 (SSE) を使用すると、 AWS Key Management Service () で管理されているキーを使用して Amazon SNS トピックのメッセージの内容を保護することにより、暗号化されたトピックに機密データを保存できますAWS KMS。

Amazon SNS が受信したメッセージはすぐに、SSE によって暗号化されます。メッセージは暗号化された形式で保存され、送信時にのみ復号されます。

- AWS Management Console または AWS SDK for Java を使用した SSE の管理 (`CreateTopic` および `SetTopicAttributes` APIアクションを使用して `KmsMasterKeyId` 属性を設定) については、「Amazon SNS トピックのサーバー側の暗号化 (SSE) を有効にする」を参照してください。
- AWS CloudFormation を使用した暗号化されたトピックの作成 (`AWS::SNS::Topic` リソースを使用して `KmsMasterKeyId` プロパティを設定) については、「AWS CloudFormation ユーザーガイド」を参照してください。

> **Important**
>
> SSE が有効なトピックへのリクエストでは必ず、HTTPS と署名バージョン 4 を使用する必要があります。
> 暗号化されたトピックとの他のサービスとの互換性については、サービスのドキュメントを参照してください。

Amazon SNS は、対称暗号化 KMS キーのみをサポートします。他のタイプの KMS キーを使用してサービスリソースを暗号化することはできません。KMS キーが対称暗号化キーかどうかを判別するには、「非対称 KMS キーを識別する」を参照してください。

AWS KMSは、安全で可用性の高いハードウェアとソフトウェアを組み合わせて、クラウド向けに拡張されたキー管理システムを提供します。AWS KMS と共に Amazon SNS を使用する場合は、メッセージデータを暗号化したデータキーも暗号化され、保護対象のデータと共に格納されます。

AWS KMSを使用する利点は次のとおりです。

- お客様自身で AWS KMS key を作成および管理できます。
- アカウントとリージョンごとに一意の AWS マネージド KMS キーを Amazon SNS に使用することもできます。
- AWS KMS のセキュリティ標準は、暗号化関連のコンプライアンス要件を満たすために役立ちます。

詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS Key Management Service とは」を参照してください。

トピック

- 暗号化スコープ
- 重要な用語

暗号化スコープ

SSE では、Amazon SNS トピック内のメッセージの本文が暗号化されます。

SSEでは、以下は暗号化されません。

- トピックのメタデータ (トピック名と属性)
- メッセージのメタデータ (件名、メッセージ ID、タイムスタンプ、属性)
- データ保護ポリシー
- トピックごとのメトリクス

> **Note**
>
> - メッセージが暗号化されるのは、トピックの暗号化が有効になった後に送信される場合のみです。Amazon SNS は、バックログされたメッセージを暗号化しません。
> - トピックの暗号化が無効になっても、暗号化されたメッセージは暗号化された状態で維持されます。

重要な用語

以下の重要なキーは、SSEの機能を理解するうえで役立ちます。詳細については、「Amazon Simple Notification Service API リファレンス」を参照してください。

データキー

データ暗号化キー (DEK) は、Amazon SNS メッセージの内容を暗号化します。

詳細については、『AWS Key Management Service デベロッパーガイド』の「データキー」と、『AWS Encryption SDK デベロッパーガイド」の「エンベロープ暗号化」を参照してください。

AWS KMS key ID

お客様アカウントまたは別のアカウントにある AWS KMS key またはカスタム AWS KMS のエイリアス、エイリアス ARN、キー ID またはキー ARN。Amazon SNS 用 AWS マネージド AWS KMS のエイリアスは常に `alias/aws/sns` ですが、カスタム AWS KMS のエイリアスは、`alias/MyAlias` のように設定できます。これらの AWS KMS キーを使用して、Amazon SNS トピック内のメッセージを保護することができます。

> **Note**
>
> 以下に留意してください。
>
> - AWS Management Console を初めて使用してトピックに Amazon SNS 用の AWS マネージド KMS を指定する場合は、AWS KMS は、Amazon SNS 用の AWS マネージド KMS を作成します。
> - または、SSE が有効な状態で、トピックで `Publish` アクションを初めて使用する場合は、AWS KMS は Amazon SNS 用 AWS マネージド KMS を作成します。

AWS KMS コンソールの [AWS KMS keys] セクションまたは `CreateKey` AWS KMS アクションを使用して、AWS KMS キーの作成、AWS KMS キーの使用方法を制御するポリシーの定義、および AWS KMS の使用状況の監査を行うことができます。詳細については、「AWS KMS keys」および「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。AWS KMS 識別子のその他の例については、 API リファレンスKeyIdの「」を参照してください。 AWS Key Management Service AWS KMS 識別子の検索については、AWS Key Management Service デベロッパーガイドの「キー ID と ARN の検索」を参照してください。

> **Important**
>
> AWS KMSを使用するための追加料金はかかります。詳細については、「AWS KMSコストの見積もり」と「AWS Key Management Service 料金表」を参照してください。

キーの管理

次のセクションでは、AWS Key Management Service (AWS KMS) で管理されるキーの操作方法について説明します。詳細について

> **Note**
>
> Amazon SNS は、対称暗号化 KMS キーのみをサポートします。他のタイプの KMS キーを使用してサービスリソースを暗号化することはできません。KMS キーが対称暗号化キーかを判別するには、「非対称 KMS キーを識別する」を参照してください。

トピック

- AWS KMSコストの見積もり
- AWS KMS 許可を設定する
- AWS KMS エラー

AWS KMSコストの見積もり

コストを予測して AWS の請求内容をより正確に把握するため、Amazon SNS が AWS KMS key を使用する頻度を調べることをお勧めします。

> **Note**
>
> コストは下の計算式でかなり正確に予測できますが、Amazon SNS の分散性により、実際のコストの方が高くなることがあります。

API リクエスト (R) トピックごとの数を計算する場合は、次の式を使用します。

```
R = B / D * (2 * P)
```

B は請求期間(秒)です。

D は、データキー再利用期間 (秒単位で、 Amazon SNS では、最大 5 分間データキーを再利用します) を表します。

P は、Amazon SNS トピックに送信するプリンシパル発行数です。

以下は計算の例です。正確な料金については、「AWS Key Management Service 料金表」を参照してください。

例 1: AWS KMS API コール数の計算 (1 公開プリンシパル、1 トピック)

この例では、以下を想定しています。

- 請求期間は1月1日から31日(2,678,400秒)です。
- データキー再利用期間は 5 分 (300 秒) です。
- 1 トピックあります。
- 公開中のプリンシパルが 1 つあります。

```
2,678,400 / 300 * (2 * 1) = 17,856
```

例 2: AWS KMS API 呼び出し数の計算 (複数の公開プリンシパル、2 トピック)

この例では、以下を想定しています。

- 請求期間は2月1日から28日(2,419,200秒) です。
- データキー再利用期間は 5 分 (300 秒) です。
- 2 トピックあります。

- 最初のトピックには、公開中のプリンシパルが 3 つあります。
- 2 つめのトピックには、公開中のプリンシパルが 5 つあります。

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

AWS KMS 許可を設定する

SSE を使用するには、トピックの暗号化およびメッセージの暗号化と復号を許可するよう AWS KMS key ポリシーを設定する必要があります。AWS KMS アクセス許可の詳細については、『AWS Key Management Service デベロッパーガイド』の「AWS KMS API アクセス許可: アクションとリソースのリファレンス」を参照してください。サーバー側の暗号化を使用して Amazon SNS トピックを設定する方法の詳細については、「サーバー側の暗号化を使用して Amazon SNS トピックをセットアップする」を参照してください。

> **Note**
>
> IAM ポリシーを使用して対称暗号化 KMS キーのアクセス許可を管理することもできます。詳細については、「AWS KMS での IAM ポリシーの使用」を参照してください。
> Amazon SNS との間の送受信のグローバルなアクセス許可を設定できますが、AWS KMS では IAM ポリシーの `Resource` セクションにおいて、特定リージョンで KMS の完全 ARN を明示的に指定することが求められます。

AWS KMS key のキーポリシーで必要な許可を付与していることも確認する必要があります。そのためには、Amazon SNS で暗号化されたメッセージを作成するプリンシパルと消費するプリンシパルをユーザーとして KMS キーポリシーで指定します。

または、Amazon SNS で暗号化されたメッセージを受け取るために発行およびサブスクライブするプリンシパルに割り当てられた IAM ポリシーで、必要な AWS KMS アクションと KMS ARN を指定できます。詳細については、AWS Key Management Service デベロッパーガイドの「AWS KMS へのアクセス管理」を参照してください。

Amazon SNS トピックのカスタマー管理キーを選択し、エイリアスを使用して IAM ポリシーまたは KMS キーポリシーと条件キー `kms:ResourceAliases` により KMS キーへのアクセスを制御している場合は、選択したカスタマー管理キーにもエイリアスが関連付けられていることを確認してください。エイリアスを使用して KMS キーへのアクセスを制御する方法の詳細については、「AWS Key Management Service デベロッパーガイド」の「エイリアスを使用して KMS キーへのアクセスを制御する」を参照してください。

SSE を使用したトピックへのメッセージの送信をユーザーに許可する

パブリッシャーには、AWS KMS key に対する kms:GenerateDataKey* および kms:Decrypt アクセス許可が必要です。

```
{
  "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"
  }]
}
```

AWS サービスと暗号化されたトピックのイベントソース間の互換性を有効化する

イベントは、複数の AWS サービスから Amazon SNS トピックに発行されます。これらのイベントソースで暗号化されたトピックを操作できるようにするには、以下のステップを実行します。

1. カスタマーマネージドキーを使用します。詳細については、AWS Key Management Service デベロッパーガイドのキーの作成を参照してください。
2. AWS のサービスにアクセス許可として kms:GenerateDataKey* と kms:Decrypt の許可を付与するには、次のステートメントを KMS ポリシーに追加します。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
```

```
    ],
    "Resource": "*"
  }]
}
```

[イベントソース]	サービスプリンシパル
Amazon CloudWatch	cloudwatch.amazonaws.com
Amazon CloudWatch Events	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

[イベントソース]	サービスプリンシパル
AWS Systems Manager Incident Manager	AWS Systems Manager Incident Manager は、次の 2 つのサービス原則で構成されています。 `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

> **Note**
>
> 一部の Amazon SNS イベントソースでは、AWS KMS key ポリシーで IAM ロール (サービスプリンシパルではありません) を指定する必要があります。
>
> - Amazon EC2 Auto Scaling
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2 Image Builder

3. KMS リソースポリシーに `aws:SourceAccount` キーと `aws:SourceArn` 条件キーを追加して、混乱した代理攻撃から KMS キーをさらに保護します。それぞれの場合の正確な詳細については、サービス固有のドキュメントリスト (上記) を参照してください。

> **Important**
>
> AWS KMS ポリシーへの `aws:SourceAccount` および `aws:SourceArn` の追加は、EventBridge-to-encrypted トピックではサポートされていません。

```
{
  "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. KMS を使用して、トピックの SSE を有効化します。
5. 暗号化されたトピックの ARN をイベントソースに追加します。

AWS KMS エラー

Amazon SNS および AWS KMS を使用する際、エラーが発生することがあります。次のリストは、エラーおよび考えられるトラブルシューティング策を示しています。

KMSAccessDeniedException

暗号化テキストは、存在しないキーまたはアクセス権限のないキーを参照しています。

HTTP ステータスコード: 400

KMSDisabledException

指定された KMS が有効ではないため、リクエストが拒否されました。

HTTP ステータスコード: 400

KMSInvalidStateException

指定されたリソースの状態がこのリクエストに対して有効ではないため、リクエストが拒否されました。詳細については、「AWS Key Management Service 開発者ガイド」の「AWS KMS keys のキーステータス」を参照してください。

HTTP ステータスコード: 400

KMSNotFoundException

指定されたエンティティまたはリソースが見つからないため、リクエストが拒否されました。

HTTP ステータスコード: 400

KMSOptInRequired

サービスを利用するためには、AWS アクセスキー ID を取得する必要があります。

HTTP ステータスコード: 403

KMSThrottlingException

リクエストは、制限が必要なために実行が拒否されました。スロットリングの詳細については、「AWS Key Management Service デベロッパーガイド」の「クォータ」を参照してください。

HTTP ステータスコード: 400

Amazon SNS トピックのサーバー側の暗号化 (SSE) を有効にする

サーバー側の暗号化 (SSE) では、機密データを暗号化されたトピックに保存できます。SSE は、AWS Key Management Service (AWS KMS) のマネージドキーを使用して、Amazon SNS トピック内のメッセージの内容を保護します。Amazon SNS でのサーバー側の暗号化の詳細については、「保管中の暗号化」を参照してください。AWS KMS キーの作成の詳細については、「AWS Key Management Service デベロッパーガイド」の「キーの作成」を参照してください。

> **Important**
>
> SSE が有効なトピックへのリクエストでは必ず、HTTPS と署名バージョン 4 を使用する必要があります。

AWS Management Console を使用して Amazon SNS トピックのサーバー側の暗号化 (SSE) を有効にする

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. [トピック] ページで、トピックを選択し、[アクション]、[編集] の順に選択します。
4. [暗号化] セクションを展開し、以下の操作を実行します。

   a. [暗号化の有効化] を選択します。

b. AWS KMS キーを指定します。詳細については、「重要な用語」を参照してください。

KMS タイプごとに、[Description] (説明)、[Account] (アカウント)、および [KMS ARN] が表示されます。

> **Important**
>
> KMS の所有者ではない場合、または `kms:ListAliases` および `kms:DescribeKey` の許可がないアカウントでログインした場合、Amazon SNS コンソールで KMS に関する情報を表示できません。
> これらの許可を付与するように、KMS の所有者へ依頼してください。詳細については、『AWS Key Management Service デベロッパーガイド』の「AWS KMS API アクセス権限: アクションとリソースのリファレンス」を参照してください。

- デフォルトでは、Amazon SNS 用の AWS マネージド CMK [(Default) alias/aws/sns] ((デフォルト) alias/aws/sns) が選択されています。

> **Note**
>
> 以下に留意してください。
>
> - AWS Management Console を初めて使用してトピックに Amazon SNS 用の AWS マネージド KMS を指定する場合は、AWS KMS は、Amazon SNS 用の AWS マネージド KMS を作成します。
> - または、SSE が有効な状態で、トピックで `Publish` アクションを初めて使用する場合は、AWS KMS は Amazon SNS 用 AWS マネージド KMS を作成します。

- AWS アカウントからカスタム KMS を使用するには、[KMS キー] フィールドを選択し、リストからカスタム KMS を選択します。

> **Note**
>
> カスタム KMS の作成手順については、AWS Key Management Service デベロッパーガイドの「キーの作成」を参照してください。

- AWS アカウントまたは別の AWS アカウントからカスタム KMS ARN を使用するには、それを [KMS キー] フィールドに入力します。

5. [Save changes] (変更の保存) をクリックします。

   トピックの SSE が有効になり、[******] ページが表示されます。

   トピックの暗号化ステータス、AWS アカウント、カスタマーマスターキー (CMK)、CMK ARN、および説明が [暗号化] タブに表示されます。

サーバー側の暗号化を使用して Amazon SNS トピックをセットアップする

KMS キーを作成するときは、次の KMS キーポリシーを使用します。

```
{
    "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"
        }
    }
}
```

暗号化された Amazon SQS キューをサブスクライブして Amazon SNS トピックのサーバー側の暗号化 (SSE) を有効にする

トピックのサーバー側の暗号化 (SSE) を有効にしてトピックのデータを保護できます。暗号化された Amazon SQS キューにメッセージを送信することを Amazon SNS に許可するには、Amazon

SQS キューに関連付けられたカスタマーマネージドキーのポリシーステートメントにより、AWS KMS API アクション (`GenerateDataKey` と `Decrypt`) へのアクセス権を Amazon SNS サービスプリンシパルに付与する必要があります。SSE の使用の詳細については、「保管中の暗号化」を参照してください。

このページでは、AWS Management Console を使用して、暗号化された Amazon SQS キューのサブスクライブ先である Amazon SNS トピックの SSE を有効にする方法について説明します。

ステップ 1: カスタム KMS キーを作成する

1. 少なくとも `AWSKeyManagementServicePowerUser` ポリシーを持つユーザーで AWS KMS コンソールにサインインします。
2. [キーの作成] を選択します。
3. 対称暗号化 KMS キーを作成するには、[Key type]] (キーのタイプ) で [Symmetric] (対称) を選択します。

   AWS KMS コンソールで非対称 KMS キーを作成する方法については、「非対称 KMS キーを作成する (コンソール)」を参照してください。
4. [Key usage] (キーの使用) では、[Encrypt and decrypt] (暗号化および復号) オプションがすでに選択されています。

   MAC コードを生成して検証する KMS キーの作成方法については、「HMAC KMS キーの作成」を参照してください。

   [詳細オプション] については、「特定用途のキー」を参照してください。
5. [Next] (次へ) をクリックします。
6. KMS キーのエイリアスを入力します。エイリアス名の先頭を **`aws/`** にすることはできません。この **`aws/`** プレフィックスは、アカウント内の AWS マネージドキー を表すために、Amazon Web Services によって予約されます。

> **Note**
>
> エイリアスを追加、削除、更新すると、KMS キーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「AWS KMS の ABAC」および「エイリアスを使用して KMS キーへのアクセスを制御する」を参照してください。

エイリアスは KMS キーを識別するために使用する表示名です。保護する予定のデータタイプ、または KMS キーで使用する予定のアプリケーションを示すエイリアスを選択することをお勧めします。

エイリアスは AWS Management Console で KMS キーを作成するときに必要です。CreateKey オペレーションを使用する場合、これらのオペレーションはオプションです。

7. (オプション) KMS キーの説明を入力します。

   今すぐ説明を追加するか、キーの状態が `Pending Deletion` または `Pending Replica Deletion` でない限り、後でいつでも更新できます。既存のカスタマーマネージドキーの説明を追加、変更、削除するには、AWS Management Console で説明を編集するか、UpdateKeyDescription オペレーションを使用します。

8. (オプション) タグキーとオプションのタグ値を入力します。KMS キーに複数のタグを追加するには、[Add tag] (タグを追加) を選択します。

   > **Note**
   >
   > KMS キーのタグ付けまたはタグ解除により、KMS キーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「AWS KMS の ABAC」および「タグを使用してKMS キーへのアクセスを制御する」を参照してください。

   AWS リソースにタグを追加すると、使用量とコストがタグごとに集計されたコスト配分レポートが AWS によって生成されます。タグは、KMS キーへのアクセスの制御にも使用できます。KMS キーのタグ付けについては、「キーのタグ付け」および「AWS KMS の ABAC」を参照してください。

9. [Next] (次へ) をクリックします。

10. KMS キーを管理できる IAM ユーザーとロールを選択します。

    > **Note**
    >
    > このキーポリシーにより、AWS アカウント はこの KMS キーを完全に制御できるようになります。これにより、アカウント管理者は IAM ポリシーを使用して、他のプリンシパルに KMS キーを管理する許可を付与できます。詳細については、「デフォルトのキーポリシー」を参照してください。

> IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

11. (オプション) 選択した IAM ユーザーとロールがこの KMS キーを削除しないようにするには、ページの下部にある [Key deletion] (キーの削除) セクションで、[Allow key administrators to delete this key] (キー管理者にこのキーの削除を許可する) のチェックボックスをオフにします。
12. [Next] (次へ) をクリックします。
13. 暗号化オペレーションでキーを使用できる IAM ユーザーとロールを選択します。[Next] (次へ) をクリックします。
14. [キーポリシーの表示と編集] ページで、次のステートメントをキーポリシーに追加し、[完了] を選択します。

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

新しいカスタマーマネージドキーがキーのリストに表示されます。

ステップ 2: 暗号化された Amazon SNS トピックを作成する

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. [Create topic] (トピックの作成) を選択します。
4. [新しいトピックの作成] ダイアログボックスの [トピック名] に、トピックの名前 (例: `MyEncryptedTopic`) を入力し、[トピックの作成] を選択します。
5. [暗号化] セクションを展開し、以下の操作を実行します。

a. [サーバー側の暗号化を有効にする] を選択します。

b. カスタマーマネージドキーを指定します。詳細については、「重要な用語」を参照してください。

カスタマーマネージドキーのタイプごとに、説明、アカウント、およびカスタマーマネージドキー ARN が表示されます。

> **Important**
>
> カスタマーマネージドキーの所有者ではない場合、または `kms:ListAliases` および `kms:DescribeKey` の許可がないアカウントでログインした場合、Amazon SNS コンソールでカスタマーマネージドキーに関する情報を表示できません。これらの許可を付与するように、カスタマーマネージドキーの所有者へ依頼してください。詳細については、『AWS Key Management Service デベロッパーガイド』の「AWS KMS API アクセス権限: アクションとリソースのリファレンス」を参照してください。

c. [カスタマーマネージドキー] で、以前に作成した [MyCustomKey] を選択してから、[サーバー側の暗号化を有効化] を選択します。

6. [Save changes] (変更の保存) をクリックします。

トピックの SSE が有効になり、[マイトピック] ページが表示されます。

トピックの暗号化ステータス、AWS アカウント、カスタマーマネージドキー、カスタマーマネージドキーの ARN、および説明が [暗号化] タブに表示されます。

暗号化された新しいトピックが、トピックのリストに表示されます。

ステップ 3: 暗号化された Amazon SQS キューを作成してサブスクライブする

1. Amazon SQS コンソールにサインインします。
2. [新しいキューの作成] を選択します。
3. [新しいキューの作成] ページで、次の操作を実行します。

a. [キュー名] (例: `MyEncryptedQueue1`) を入力します。

b. [標準キュー]、[キューの設定] の順に選択します。

c. [SSE の使用] を選択します。

d. [AWS KMS key] で、以前に作成した [MyCustomKey] を選択してから、[キューの作成] を選択します。

4. プロセスを繰り返して、2 つめのキュー (例えば、`MyEncryptedQueue2`) を作成します。

   暗号化された新しいキューがキューのリストに表示されます。

5. Amazon SQS コンソールで、`MyEncryptedQueue1` および `MyEncryptedQueue2` を選択して、[キューのアクション]、[SNS トピックへのキューのサブスクライブ] の順に選択します。

6. [トピックへのサブスクライブ] ダイアログボックスの [トピックの選択] で、[MyEncryptedTopic]、[サブスクライブ] の順に選択します。

   暗号化されたトピックに対する暗号化されたキューのサブスクリプションは、[トピックのサブスクリプション結果] ダイアログボックスに表示されます。

7. [OK] をクリックします。

ステップ 4: 暗号化されたトピックにメッセージを発行する

1. Amazon SNS コンソールにサインインします。
2. ナビゲーションパネルで、[トピック] を選択します。
3. トピックのリストから [MyEncryptedTopic] を選択し、[メッセージの発行] を選択します。
4. [メッセージの発行] ページで、次の操作を行います。

   a. (オプション) [メッセージの詳細] セクションで、[件名] (`Testing message publishing` など) を入力します。

   b. [メッセージの本文] セクションで、メッセージの本文 (`My message body is encrypted at rest.` など) を入力します。

   c. [メッセージの発行] を選択します。

メッセージは、サブスクライブされた暗号化キューに発行されます。

ステップ 5: メッセージの配信を確認する

1. Amazon SQSコンソールにサインインします。
2. キューのリストから [MyEncryptedQueue1] を選択し、次に [Send and receive messages] (メッセージの送受信) を選択します。

3. [Send and receive messages in MyEncryptedQueue1] (MyEncryptedQueue1 のメッセージの送信と受信) ページで、[Poll for messages] (メッセージのポーリング) を選択します。

   先ほど送信したメッセージが表示されます。
4. メッセージを表示するには、[詳細] を選択します。
5. 完了したら、[閉じる] をクリックします。
6. [MyEncryptedQueue2] に対してこの処理を繰り返します。

インターネットトラフィックのプライバシー

Amazon SNS の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントは、Amazon SNS への接続のみを許可する VPC 内の論理エンティティです。VPC はリクエストを Amazon SNS にルーティングし、レスポンスを VPC にルーティングします。以下のセクションでは、VPC エンドポイントの使用と VPC エンドポイントポリシーの作成について説明します。

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合、VPC と Amazon SNS の間のプライベート接続を確立できます。この接続では、メッセージをパブリックインターネット経由で送信せずに、Amazon SNS トピックに発行できます。

Amazon VPC は、お客様の定義する仮想ネットワークで AWS リソースを起動するために使用できる AWS のサービスです。VPC を使用すると、IP アドレス範囲、サブネット、ルートテーブル、ネットワークゲートウェイなどのネットワーク設定を制御できます。VPC を Amazon SNS に接続するには、インターフェイス VPC エンドポイントを定義します。このタイプのエンドポイントにより、VPC を AWS サービスに接続できるようになります。このエンドポイントは、インターネットゲートウェイ、ネットワークアドレス変換 (NAT) インスタンス、または VPN 接続を必要とせず、信頼性が高くスケーラブルな Amazon SNS への接続を提供します。詳細については、『Amazon VPC ユーザーガイド』の「インターフェイス VPC エンドポイント」を参照してください。

このセクションの情報は、Amazon VPC のユーザーを対象にしています。VPC の詳細を確認したり、作成を開始したりするには、『Amazon VPC ユーザーガイド』の「Amazon VPC の開始方法」を参照してください。

> **Note**
>
> VPC エンドポイントでは、Amazon SNS トピックをプライベート IP アドレスにサブスクライブすることはできません。

トピック

- Amazon SNS 用の Amazon VPC エンドポイントの作成
- Amazon SNS 用の VPC エンドポイントポリシーを作成する
- Amazon VPC から Amazon SNS メッセージを発行する

Amazon SNS 用の Amazon VPC エンドポイントの作成

Amazon VPC から Amazon SNS トピックにメッセージを発行するには、インターフェイス VPC エンドポイントを作成します。次に、VPC で管理するネットワーク内でトラフィックを維持しながら、トピックにメッセージを発行できます。

次の情報を使用してエンドポイントを作成し、VPC と Amazon SNS 間の接続をテストします。一から開始する作業に役立つチュートリアルについては、「Amazon VPC から Amazon SNS メッセージを発行する」を参照してください。

エンドポイントの作成

、、 AWS SDK、Amazon SNS API AWS Management Console、またはを使用して VPC に Amazon SNS エンドポイントを作成できます。 AWS CLI AWS CloudFormation

Amazon VPC コンソールまたは AWS CLIを使用して、エンドポイントを作成および設定する方法については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントの作成」を参照してください。

> **Important**
>
> Amazon Virtual Private Cloud は HTTPS Amazon SNS エンドポイントでのみ使用できます。エンドポイントを作成するとき、VPC の接続先のサービスとして Amazon SNS を指定します。Amazon VPC コンソールで、このサービス名は選択したリージョンによって異なります。例えば、米国東部 (バージニア北部) を選択した場合、サービス名は com.amazonaws.us-east-1.statesになります。
> Amazon VPC からメッセージを送信するように Amazon SNS を設定する場合、プライベート DNS を有効にして、`sns.us-east-2.amazonaws.com` の形式でエンドポイントを指定する必要があります。
> プライベートDNS は、`queue.amazonaws.com`や`us-east-2.queue.amazonaws.com`などのレガシーエンドポイントをサポートしていません。

を使用してエンドポイントを作成および設定する方法については AWS CloudFormation、*AWS::EC2::VPCEndpoint*ユーザーガイドのリソースを参照してください。AWS CloudFormation

VPC と Amazon SNS との間の接続をテストする

Amazon SNS のエンドポイントを作成したら、VPC から Amazon SNS トピックにメッセージを発行できます。この接続をテストするには、以下の手順を実行します。

1. VPC にある Amazon EC2 インスタンスに接続します。接続の詳細については、Amazon EC2 ドキュメントの Linux インスタンスへの接続または Windows インスタンスへの接続を参照してください。

   例えば、SSH クライアントを使用して Linux インスタンスに接続するには、ターミナルから以下のコマンドを実行します。

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   実行する条件は以下の通りです。

   - ec2-key-pair.pem は、インスタンスの作成時に Amazon EC2 から提供されたキーペアを含むファイルです。
   - instance-hostname は、インスタンスのパブリックホスト名です。Amazon EC2 コンソールでホスト名を取得するには、[インスタンス] を選択してから、インスタンスを選択して、[パブリック DNS (IPv4)] の値を見つけます。
2. インスタンスから、 AWS CLIで Amazon SNS `publish` コマンドを使用します。以下のコマンドを使用して、シンプルなメッセージをトピックに送信できます。

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
   ```

   コードの説明は以下のとおりです。

   - AWS aws-region は、トピックが配置されているリージョンです。
   - sns-topic-arnトピックの Amazon リソースネーム (ARN) です。Amazon SNS コンソールから ARN を取得するには: [トピック] を選択し、自分のトピックを検索して、[ARN] カラムでその値を見つけます。

メッセージが Amazon SNS によって正常に受信された場合、ターミナルには以下のようなメッセージ ID が表示されます。

```
{
   "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

Amazon SNS 用の VPC エンドポイントポリシーを作成する

Amazon SNS の Amazon VPC エンドポイントに対するポリシーを作成して、以下を指定することができます。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、『Amazon VPC ユーザーガイド』の「VPC エンドポイントでサービスへのアクセスを制御する」を参照してください。

以下の例の VPC エンドポイントポリシーでは、Amazon SNS トピック `MyTopic` に対する発行を IAM ユーザー `MyUser` に許可することを指定します。

```
{
  "Statement": [{
    "Action": ["sns:Publish"],
    "Effect": "Allow",
    "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
    "Principal": {
      "AWS": "arn:aws:iam:123456789012:user/MyUser"
    }
  }]
}
```

以下は拒否されます。

- 他の Amazon SNS API アクション (`sns:Subscribe` や `sns:Unsubscribe` など)。
- この VPC エンドポイントを使用しようとする IAM の他のユーザーおよびルール。

- MyUser別の Amazon SNS トピックに発行する 。

> **Note**
>
> IAM ユーザーはさらに、他の Amazon SNS API アクションを VPC の外側 から使用します。

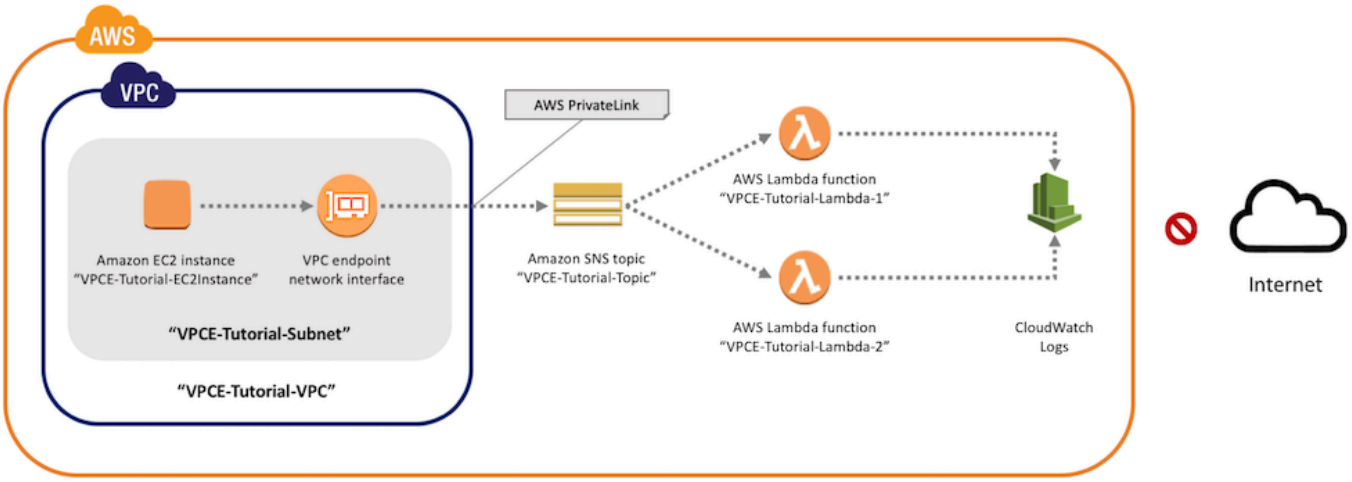
Amazon VPC から Amazon SNS メッセージを発行する

このセクションでは、プライベートネットワークでメッセージを安全に保ちながら、Amazon SNS トピックに発行する方法について説明します。Amazon Virtual Private Cloud (Amazon VPC) でホストされている Amazon EC2 インスタンスからメッセージを発行します。メッセージは、パブリックインターネットを移動せずに AWS ネットワーク内に留まります。VPC からプライベートにメッセージを発行することにより、アプリケーションと Amazon SNS 間のトラフィックのセキュリティを強化できます。このセキュリティは、顧客に関する個人を特定できる情報 (PII) を公開する場合や、アプリケーションが市場規制の対象となる場合に重要になります。例えば、プライベートな発行は、Health Insurance Portability and Accountability Act (HIPAA) に準拠する必要があるヘルスケアシステムや、Payment Card Industry Data Security Standard (PCI DSS) に準拠する必要がある財務システムがある場合に役立ちます。

一般的なステップは次のとおりです。

- AWS CloudFormation テンプレートを使用して、AWS アカウント アカウントの一時的なプライベートネットワークを自動的に作成します。
- VPC を Amazon SNS に接続する VPC エンドポイントを作成します。
- Amazon EC2 インスタンスにログインし、Amazon SNS トピックに対してプライベートにメッセージを発行します。
- メッセージが正常に配信されたことを確認します。
- このプロセス中に作成したリソースを削除し、AWS アカウント アカウントに残らないようにします。

次の図は、このステップを完了する際に AWS アカウントで作成するプライベートネットワークを示しています。



このネットワークは、Amazon EC2 インスタンスを含む VPC で構成されます。インスタンスは、インターフェイス VPC エンドポイントを介して Amazon SNS に接続します。このタイプのエンドポイントは、AWS PrivateLink を使用するサービスに接続します。この接続が確立されたら、ネットワークがパブリックインターネットから切断されている場合でも、Amazon EC2 インスタンスにログインして Amazon SNS トピックにメッセージを発行できます。このトピックは、受け取ったメッセージを、2 つの AWS Lambda サブスクライブ関数にファンアウトします。これらの関数は、受け取ったメッセージを Amazon CloudWatch Logs に記録します。

このステップの完了には 20 分ほどかかります。

トピック

開始する前に

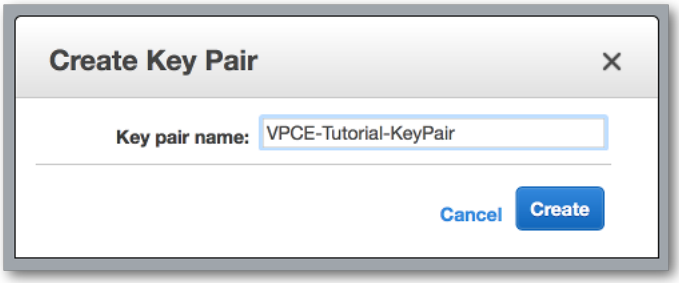
開始する前に、Amazon Web Services (AWS) アカウントが必要です。サインアップすると、Amazon SNS および Amazon VPC を含む AWS のすべてのサービスに対してお客様のアカウントが自動的にサインアップされます。アカウントをまだ作成していない場合は、https://aws.amazon.com/ に移動し、[まずは無料で始める] を選択します。

ステップ 1: Amazon EC2 キーペアを作成する

Amazon EC2 インスタンスへのログインには、キーペアが使用されます。これは、ログイン情報の暗号化に使用されるパブリックキーと、その復号に使用されるプライベートキーで構成されます。キーペアを作成するときは、プライベートキーのコピーをダウンロードします。後で、キーペアを使用して Amazon EC2 インスタンスにログインします。ログインするには、キーペアの名前を指定し、プライベートキーを指定します。

キーペアを作成するには

1. AWS Management Console にサインインし、Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. 左のナビゲーションメニューで、[ネットワーク & セキュリティ] セクションを見つけます。次に、[キーペア] を選択します。
3. [キーペアの作成] を選択します。
4. [キーペアの作成] ウィンドウで、[キーペア名] に「**VPCE-Tutorial-KeyPair**」と入力します。続いて、[作成] を選択します。



5. ブラウザによってプライベートキーファイルが自動的にダウンロードされます。これを安全な場所に保存します。Amazon EC2 により、ファイルに拡張子 .pem が付けられます。
6. (オプション) Mac または Linux コンピュータで SSH クライアントを使用してインスタンスに接続している場合は、chmod コマンドを使用してプライベートキーファイルの権限を設定すると、お客様のみがそれを読み取ることができます。

   a. ターミナルを開き、プライベートキーを含むディレクトリに移動します。

```
$ cd /filepath_to_private_key/
```

b. 次のコマンドを使用してアクセス権限を設定します。

```
$ chmod 400 VPCE-Tutorial-KeyPair.pem
```

ステップ 2: AWS リソースを作成する

インフラストラクチャをセットアップするには、AWS CloudFormation テンプレートを使用します。テンプレートは、Amazon EC2 インスタンスや Amazon SNS トピックなどの AWS リソースを構築するための設計図として機能するファイルです。このプロセスのテンプレートは GitHub で提供されていて、ダウンロードできます。

AWS CloudFormation にテンプレートを提供すると、AWS CloudFormation は AWS アカウント でスタックとして必要なリソースをプロビジョニングします。スタックは、単一のユニットとして管理できるリソースのコレクションです。このステップを終了すると、AWS CloudFormation を使用してスタック内のすべてのリソースを一度に削除できます。これらのリソースは、希望しない限り AWS アカウント アカウントには残りません。

このプロセスのスタックには、次のリソースが含まれます。

- VPC および関連するネットワーキングリソース (サブネット、セキュリティグループ、インターネットゲートウェイ、およびルートテーブルを含む)。
- VPC 内のサブネットに起動された Amazon EC2 インスタンス。
- Amazon SNS トピック
- 2 つの AWS Lambda 関数。これらの関数は、Amazon SNS トピックに発行されたメッセージを受け取り、CloudWatch Logs にイベントを記録します。
- Amazon CloudWatch メトリクスおよびログ
- Amazon EC2 インスタンスに Amazon SNSの使用を許可する IAM ロール、および CloudWatch Logs への書き込みを Lambda 関数に許可する IAM ロール。

AWS リソースを作成するには

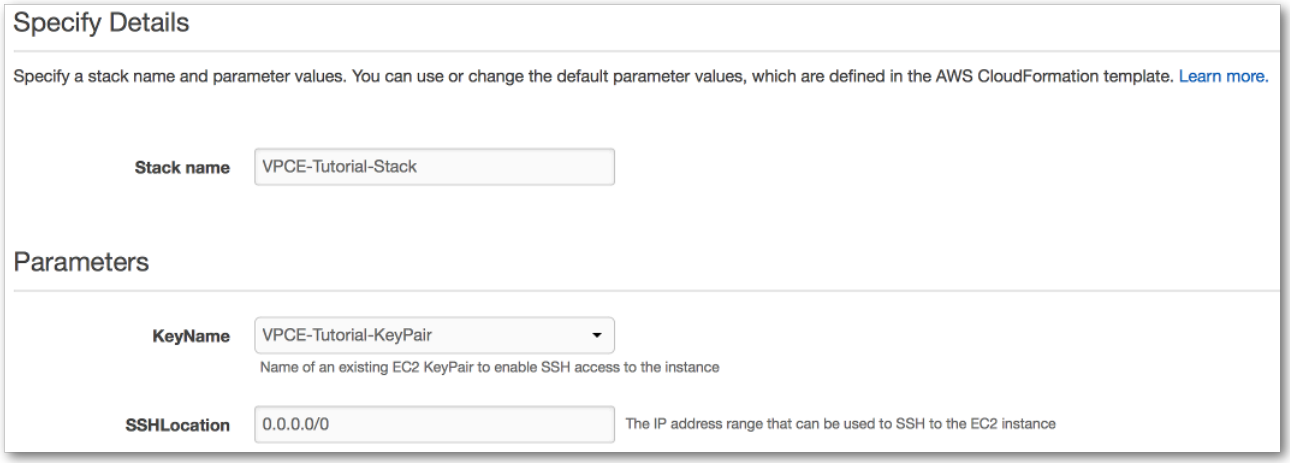
1. GitHub ウェブサイトからテンプレート ファイルをダウンロードします。
2. AWS CloudFormation コンソールにサインインします。

3. [スタックの作成] を選択します。
4. [テンプレートの選択] ページで、[テンプレートを Amazon S3 にアップロード] を選択してから、ファイルを選択して [次へ] をクリックします。
5. [詳細の指定] ページで、スタック名とキー名を指定します。

   a. [スタックの名前] に **`VPCE-Tutorial-Stack`** を入力します。

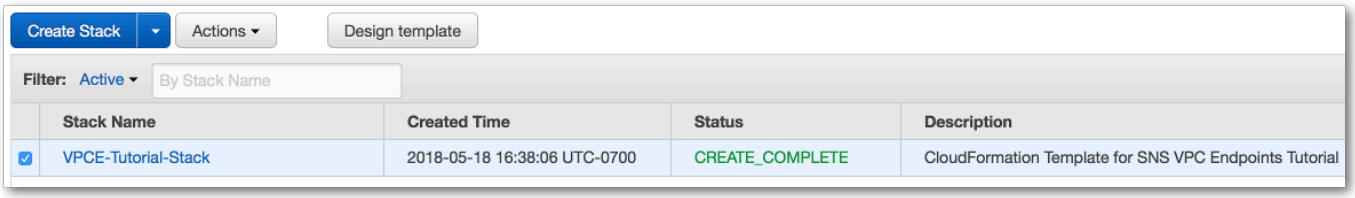
   b. [KeyName] で、[VPCE-Tutorial-KeyPair] を選択します。

   c. [SSHLocation] で、デフォルト値の **`0.0.0.0/0`** のままにします。



   d. [Next (次へ)] を選択します。
6. [オプション] ページで、すべてのデフォルト値を受け入れ、[次へ] を選択します。
7. [確認] ページで、スタックの詳細を確認します。
8. [機能] で、AWS CloudFormation がカスタム名で IAM リソースを作成する可能性があることに同意します。
9. [Create] を選択します。

   AWS CloudFormation コンソールで [スタック] ページが開きます。VPCE-Tutorial-Stack のステータスは CREATE_IN_PROGRESS です。数分後に作成プロセスが完了し、ステータスが CREATE_COMPLETE に変わります。



> **Tip**
>
> [更新] ボタンを選択して、スタックの最新のステータスを表示します。

ステップ 3: Amazon EC2 インスタンスにインターネットアクセスがないことを確認する

前のステップの VPC で起動された Amazon EC2 インスタンスにはインターネットアクセスがありません。アウトバウンドトラフィックが禁止されているため、Amazon SNS にメッセージを発行することができません。インスタンスにログインしてこれを確認します。次に、パブリックエンドポイントへの接続と、Amazon SNS へのメッセージの発行を試みます。

この時点では、発行の試みは失敗します。後のステップで、Amazon SNS の VPC エンドポイントを作成すると、発行の試みは成功します。

Amazon EC2 インスタンスに接続します。

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. 左のナビゲーションメニューで、[インスタンス] セクションを見つけます。続いて、[インスタンス] を選択します。
3. インスタンスの一覧で、[VPCE-Tutorial-EC2Instance] を選択します。
4. [Public DNS (IPv4)] カラムに示されたホスト名をコピーします。

5. ターミナルを開きます。キーペアが含まれているディレクトリから、以下のコマンドを使用してインスタンスに接続します。ここで、*instance-hostname* は、Amazon EC2 コンソールからコピーしたホスト名です。

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

インスタンスがインターネットに接続されていないことを確認するには

- ターミナルで、amazon.com など任意のパブリックエンドポイントへの接続を試します。

```
$ ping amazon.com
```

接続の試行が失敗するため、いつでもキャンセルできます (Windows では Ctrl + C、macOS では Command + C)。

インスタンスが Amazon SNS に接続されていないことを確認するには

1. Amazon SNS コンソールにサインインします。
2. 左側のナビゲーションメニューで、[トピック] を選択します。
3. [トピック] ページで、トピック [VPCE-Tutorial-Topic] の Amazon リソースネーム (ARN) をコピーします。
4. ターミナルで、トピックへのメッセージを発行を試みます。

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

発行の試みが失敗するため、いつでもキャンセルできます。

ステップ 4: Amazon SNSの Amazon VPC エンドポイントを作成する

VPC を Amazon SNS に接続するには、インターフェイス VPC エンドポイントを定義します。エンドポイントを追加したら、VPC の Amazon EC2 インスタンスにログインし、そこから Amazon SNS API を使用できます。トピックにメッセージを発行でき、メッセージはプライベートに発行されます。メッセージは AWS ネットワーク内にとどまり、パブリックインターネットを移動することはありません。

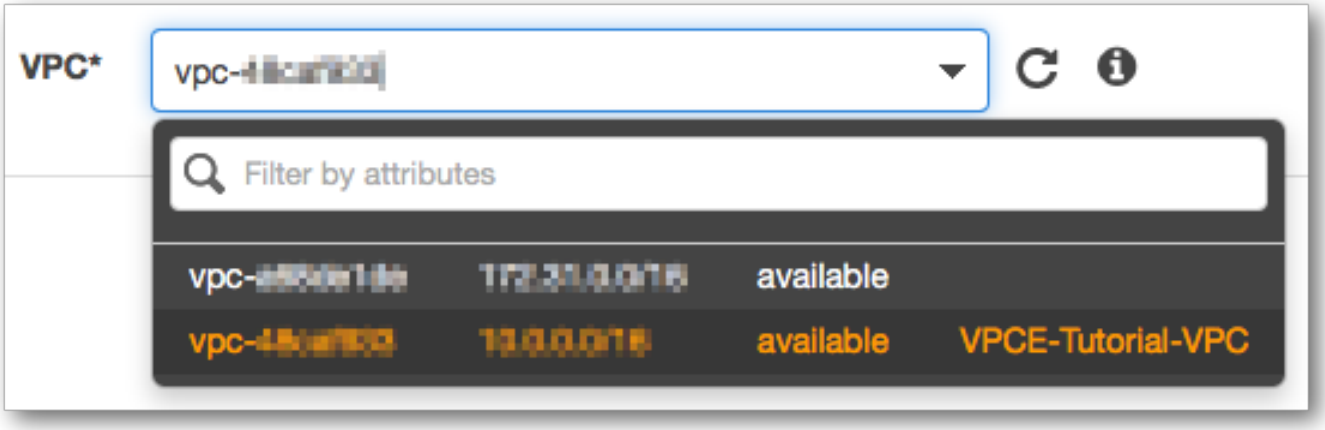
> **Note**
>
> インスタンスは、依然として AWS の他のサービスやインターネットのエンドポイントにアクセスできません。

エンドポイントを作成するには

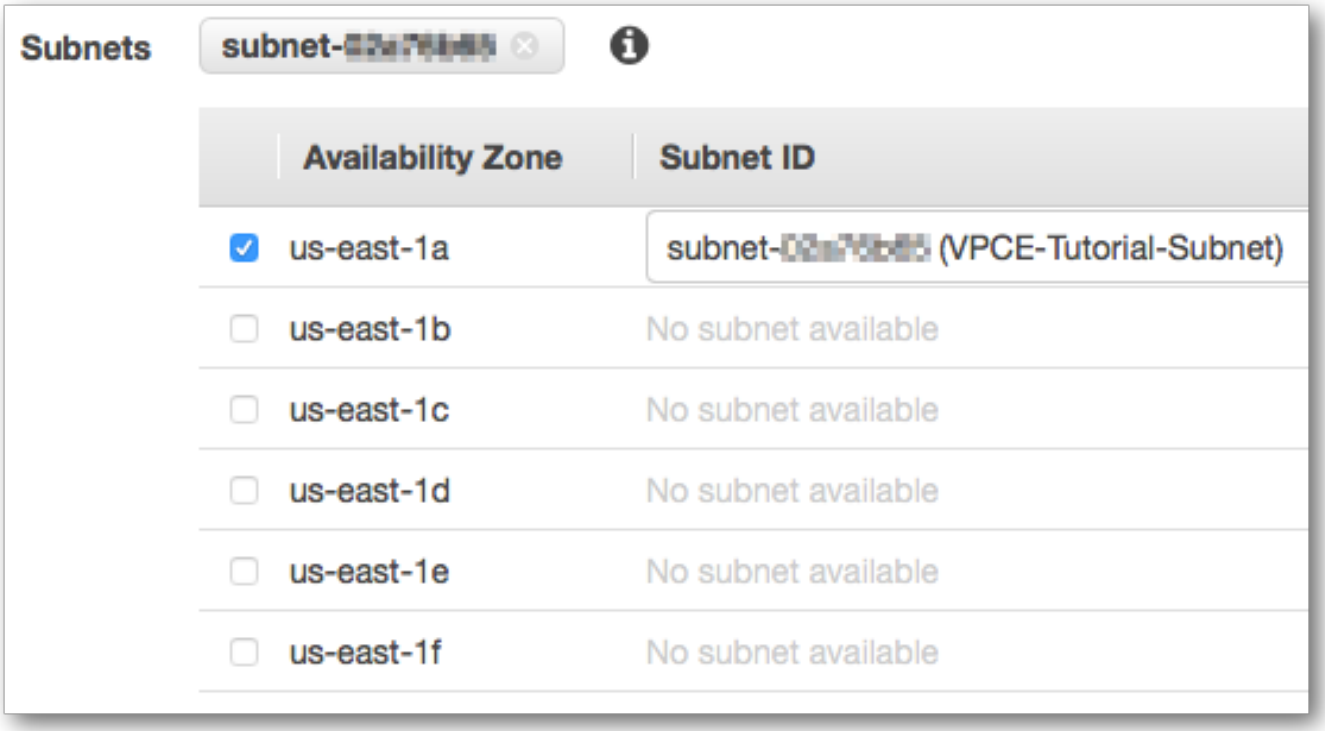
1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. 左側のナビゲーションメニューで、[エンドポイント] を選択します。
3. [エンドポイントの作成] を選択します。

4. [エンドポイントの作成] ページの [サービスカテゴリ] で、デフォルトの選択である [AWS サービス] をそのままにします。
5. [サービス名] で、Amazon SNS のサービス名を選択します。

   このサービス名は、選択したリージョンによって異なります。例えば、米国東部 (バージニア北部) を選択した場合、サービス名は com.amazonaws.***us-east-1***.snsになります。
6. [VPC] で、[VPCE-Tutorial-VPC] と言う名前の VPC を選択します。

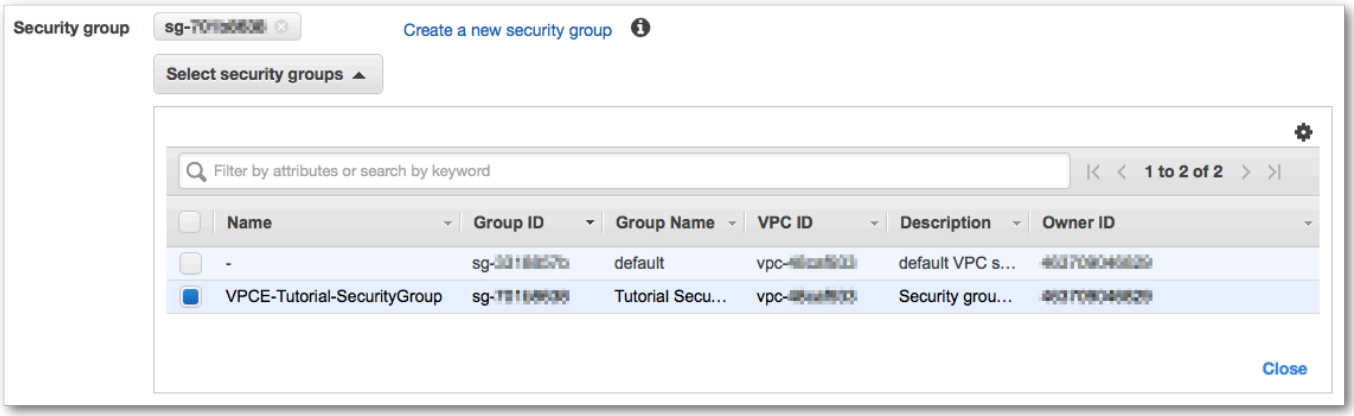


7. [サブネット] で、サブネット ID に VPCE-Tutorial-Subnet を持つサブネットを選択します。

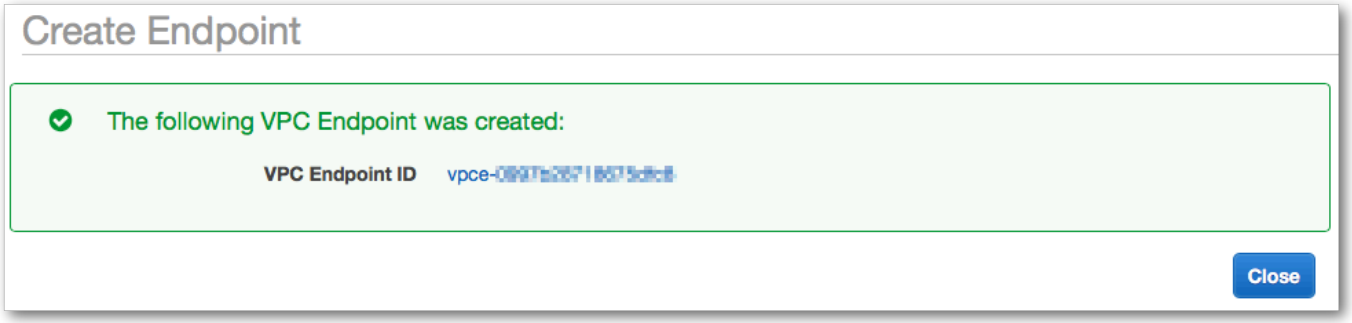


8. [プライベート DNS 名を有効にする] で、[このエンドポイントで有効にする] を選択します。

9. [セキュリティグループ] で、[セキュリティグループの選択] を選択し、[VPCE-Tutorial-SecurityGroup] を選択します。

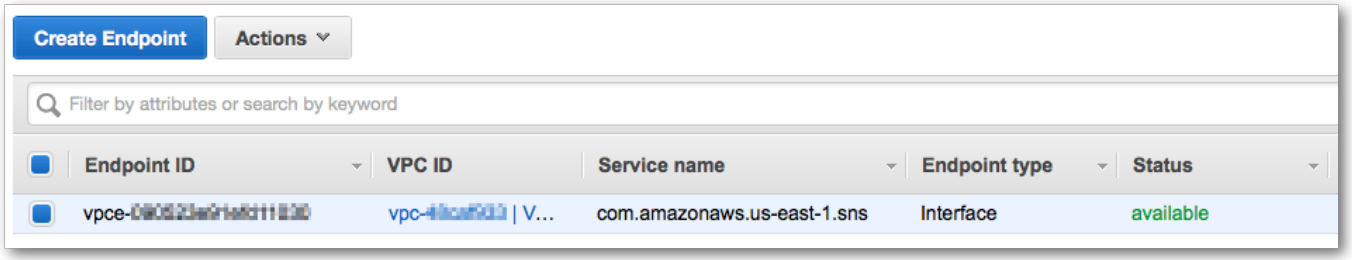


10. [エンドポイントの作成] を選択します。Amazon VPC コンソールで、VPC エンドポイントが作成されたことが確認されます。



11. [Close] を選択します。

    Amazon VPC コンソールの [エンドポイント] ページを開きます。新しいエンドポイントのステータスは [保留中] です。数分で、作成プロセスが完了すると、ステータスが [利用可能] に変わります。



ステップ 5: Amazon SNS トピックにメッセージを発行する

これで VPC に Amazon SNS のエンドポイントが含まれたので、Amazon EC2 インスタンスにログインし、トピックにメッセージを発行できます。

メッセージを発行するには

1. ターミナルが Amazon EC2 インスタンスに接続されていない場合は、再度接続します。

   ```
   $ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
   ```

2. 以前の手順と同じコマンドを実行して、Amazon SNS トピックにメッセージを発行します。今回は、発行の試みが成功すると、Amazon SNS はメッセージ ID を返します。

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"

   {
       "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
   }
   ```

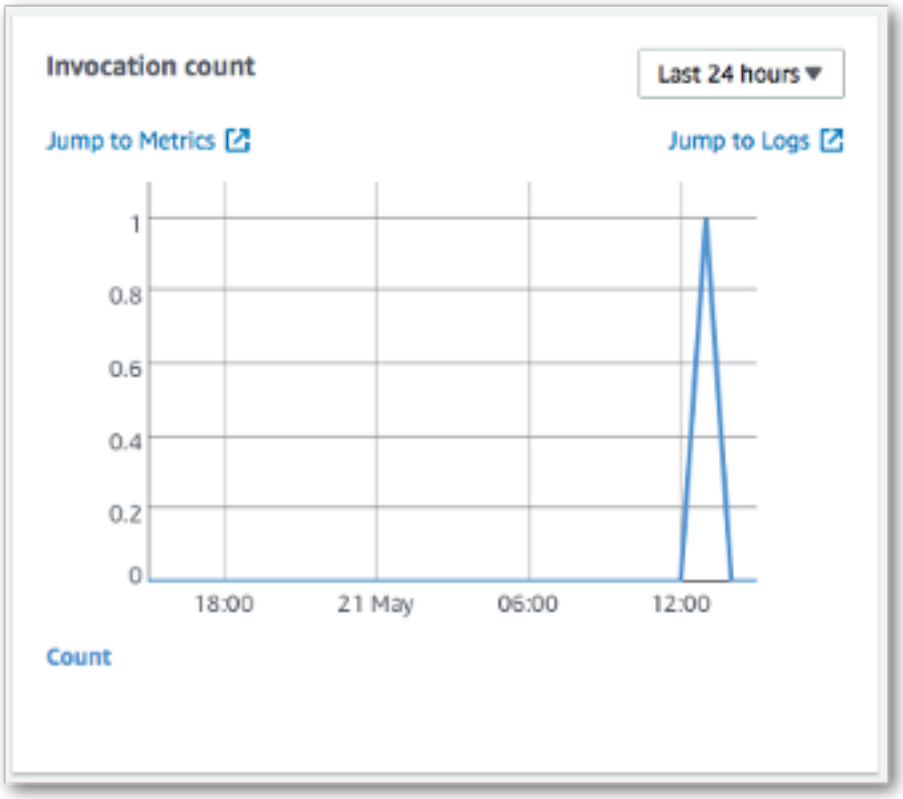
ステップ 6: メッセージの配信を確認する

Amazon SNS トピックがメッセージを受け取ると、2 つの Lambda サブスクライブ関数に送信して、メッセージをファンアウトします。これらの関数がメッセージを受け取ると、イベントを CloudWatch Logs に記録します。メッセージの配信が成功したことを確認するには、関数が呼び出されたこと、および CloudWatch Logs が更新されたことを確認します。

Lambda 関数が呼び出されたことを確認するには

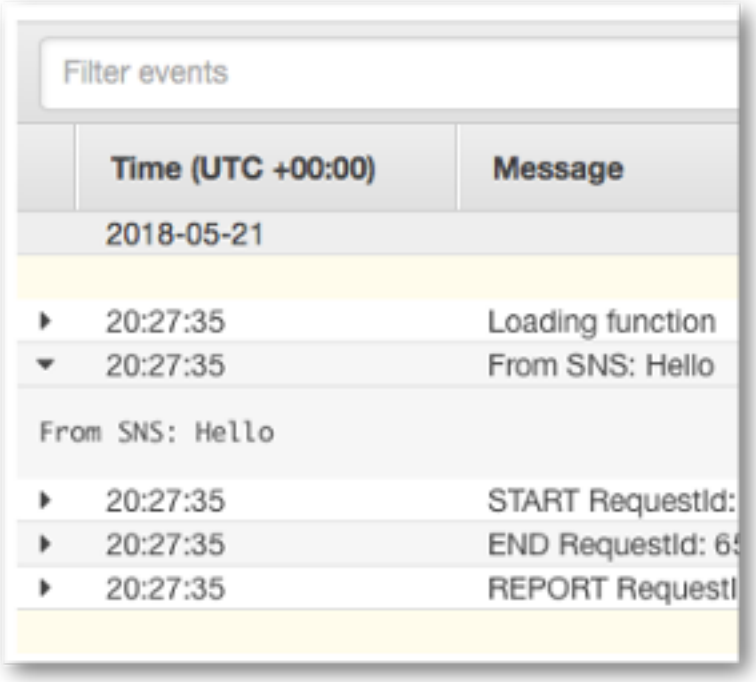
1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. [関数] ページで、[VPCE-Tutorial-Lambda-1] を選択します。
3. [モニタリング] を選択します。
4. [呼び出しカウント] グラフを確認します。このグラフには、Lambda 関数が実行された回数が表示されます。

   呼び出しカウントは、トピックにメッセージを発行した回数に一致します。



CloudWatch Logs が更新されたことを確認するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左側のナビゲーションメニューで [ログ] を選択します。
3. Lambda 関数によって書き込まれたログを確認します。
   a. [/aws/lambda/VPCE-Tutorial-Lambda-1/] ロググループを選択します。
   b. ログストリームを選択します。
   c. ログにエントリ `From SNS: Hello` が含まれていることを確認します。



d. コンソール上部の [ロググループ] を選択して、[ロググループ] ページを表示します。次に、/aws/lambda/VPCE-Tutorial-Lambda-2/ ロググループに対して前のステップを繰り返します。

おめでとうございます。Amazon SNS のエンドポイントを VPC に追加することにより、VPC で管理されるネットワーク内から、トピックにメッセージを発行することができました。メッセージは、パブリックインターネットに公開されることなくプライベートで発行されました。

ステップ 7: クリーンアップ

作成したリソースは、保持することを希望しない限り、今すぐ削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウント アカウントに請求される料金の発生を防ぎます。

まず、Amazon VPC コンソールを使用して VPC エンドポイントを削除します。次に、AWS CloudFormation コンソールでスタックを削除して、作成した他のリソースを削除します。スタックを削除すると、AWS CloudFormation によって AWS アカウント からスタックのリソースが削除されます。

VPC エンドポイントを削除するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。

2. 左側のナビゲーションメニューで、[エンドポイント] を選択します。
3. 作成したエンドポイントを選択します。
4. [アクション] を選択してから、[エンドポイントの削除] を選択します。
5. [エンドポイントの削除] ウィンドウで、[はい、削除します] を選択します。

   エンドポイントのステータスが [削除中] に変わります。削除が完了すると、エンドポイントがページから削除されます。

AWS CloudFormation スタックを削除するには

1. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソール を開きます。
2. [VPCE-Tutorial-Stack] スタックを選択します。
3. [アクション] を選択してから、[スタックの削除] を選択します。
4. [スタックの削除] ウィンドウで、[はい、削除します] を選択します。

   スタックのステータスが DELETE_IN_PROGRESS に変わります。削除が完了すると、スタックがページから削除されます。

関連リソース

詳細については、以下のリソースを参照してください。

- AWS セキュリティブログ: AWS PrivateLink で Amazon SNS に発行したメッセージを保護する
- Amazon VPC とは
- VPC エンドポイント ()
- Amazon EC2 とは
- AWS CloudFormation の概念

メッセージデータ保護セキュリティ

- メッセージデータ保護は Amazon SNS の機能で、保存中のデータとは対照的に、移動中のデータのコンテンツを監査および管理するための独自のルールとポリシーを定義するために使用されます。

- メッセージデータ保護は、メッセージ中心のエンタープライズアプリケーションにガバナンス、コンプライアンス、監査サービスを提供するため、Amazon SNS トピックの所有者は、データイングレスおよびエグレスを制御し、コンテンツフローを追跡して記録できます。
- ペイロードベースのガバナンスルールを記述して、不正なペイロードコンテンツがメッセージストリームに入るのを防ぐことができます。
- 個々のサブスクライバーに異なるコンテンツアクセス権限を付与し、コンテンツフロープロセス全体を監査できます。

Amazon SNS での Identity and Access Management

Amazon SNS へのアクセスには、AWS によってリクエストの認証に使用される認証情報が必要です。これらの認証情報には、Amazon SNS トピックやメッセージなどの AWS リソースへのアクセス権限が必要です。次のセクションでは、AWS Identity and Access Management (IAM) と Amazon SNS を使用して、リソースにアクセスできるユーザーを制御することで、リソースをセキュリティで保護する方法について詳しく説明します。

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービス です。IAM 管理者は、誰を認証 (サインインを許可) し、誰に Amazon SNS リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加費用なしで使用できる AWS のサービスです。

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon SNS で行う作業に応じて異なります。

サービスユーザー - ジョブを実行するために Amazon SNS サービスを使用する場合は、管理者から必要なアクセス許可と認証情報が与えられます。さらに多くの Amazon SNS 機能を使用して作業を行うには、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。Amazon SNS の機能にアクセスできない場合は、「Amazon Simple Notification Service アイデンティティとアクセスのトラブルシューティング」を参照してください。

サービス管理者 - 社内の Amazon SNS リソースを担当している場合は、通常、Amazon SNS へのフルアクセスがあります。サービスのユーザーがどの Amazon SNS 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解し

てください。会社で Amazon SNS と IAM を併用する方法の詳細については、「Amazon Simple Notification Service で IAM を使用する方法」を参照してください。

IAM 管理者 - 管理者は、Amazon SNS へのアクセス権を管理するポリシーの作成方法の詳細について確認する場合があります。IAM で使用可能な、Amazon SNS アイデンティティベースのポリシーの例を確認するには、「Amazon Simple Notification Service のアイデンティティベースのポリシーの例」を参照してください。

アイデンティティによる認証

認証とは、アイデンティティ認証情報を使用して AWSにサインインする方法です。ユーザーは、AWS アカウント のルートユーザーもしくは IAM ユーザーとして、または IAM ロールを引き受けることによって、認証を受ける (AWS にサインインする) 必要があります。

ID ソースから提供された認証情報を使用して、フェデレーティッドアイデンティティとして AWS にサインインできます。 AWS IAM Identity Centerフェデレーティッドアイデンティティの例としては、IAM アイデンティティセンターユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報などがあります。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWSにアクセスする場合、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。AWS へのサインインの詳細については、『AWS サインイン ユーザーガイド』の「AWS アカウント にサインインする方法」を参照してください。

プログラムで AWS にアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報でリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、『IAM ユーザーガイド』の「AWS API リクエストの署名」を参照してください。

使用する認証方法を問わず、追加のセキュリティ情報の提供が求められる場合もあります。例えば、AWS では多要素認証 (MFA) を使用してアカウントのセキュリティを高めることを推奨しています。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」 および「IAM ユーザーガイド」の「AWS での多要素認証 (MFA) の使用」を参照してください。

AWS アカウントのルートユーザー

AWS アカウント を作成する場合、このアカウントのすべての AWS のサービス とリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。このアイデンティティ

は AWS アカウントのルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報を保護し、それらを使用してルートユーザーのみが実行できるタスクを実行してください。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッド ID

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに対し、ID プロバイダーとのフェデレーションを使用して、一時的な認証情報の使用により、AWS のサービスへのアクセスを要求します。

フェデレーテッドアイデンティティ は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリのユーザーか、または ID ソースから提供された認証情報を使用して AWS のサービス にアクセスするユーザーです。フェデレーティッド ID が AWS アカウント にアクセスすると、ロールが継承され、そのロールが一時的な認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Center を使用することをお勧めします。IAM アイデンティティセンターでユーザーとグループを作成するか、すべての AWS アカウント とアプリケーションで使用するために、独自の ID ソースで一連のユーザーとグループに接続して同期することもできます。IAM アイデンティティセンター の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM アイデンティティセンター?」(IAM アイデンティティセンター とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザー は、1 人のユーザーまたは 1 つのアプリケーションに対して特定の許可を持つ AWS アカウント 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。

例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロール は、特定の許可を持つ、AWS アカウント 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロールを切り替えることにより、AWS Management Console で一時的に IAM ロールを引き受けることができます。ロールを引き受けるには、AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次のような状況で役立ちます。

- フェデレーティッドユーザーアクセス - フェデレーティッドアイデンティティに許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「サードパーティアイデンティティプロバイダー用のロールの作成」を参照してください。IAM Identity Center を使用する場合、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM アイデンティティセンターは、アクセス許可セットを IAM のロールに関連付けます。権限セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「権限セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースへのアクセスを別のアカウントの人物 (信頼できるプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

- クロスサービスアクセス - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービス に権限を委任するロールの作成」を参照してください。
  - サービスにリンクされたロール - サービスにリンクされたロールは、AWS のサービス にリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの許可を表示できますが、編集はできません。
- Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセス権の管理

AWS でアクセスを制御するには、ポリシーを作成して AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらのアクセス許可を定義します。AWS は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWSに保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は AWSJSON ポリシーを使用して、だれが何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するためのアクセス許可をユーザーに付与するため、IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWS API からロール情報を取得できます。

アイデンティティベースポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

アイデンティティベースのポリシーは、さらに インラインポリシー または マネージドポリシー に分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。管理ポリシーは、AWS アカウント 内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーとカスタマー管理ポリシーがあります。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーの例には、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは IAM の AWSマネージドポリシーは使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与される最大の許可を設定できます。

- 権限の境界 - 権限の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティに権限の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとその権限の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、権限の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。権限の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティの権限の境界」を参照してください。
- サービスコントロールポリシー (SCP) - SCP は、AWS Organizations で組織や組織単位 (OU) の最大許可を指定する JSON ポリシーです。AWS Organizations は、ユーザーのビジネスが所有する複数の AWS アカウント をグループ化し、一元的に管理するサービスです。組織内のすべ

ての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP ではメンバーアカウントのエンティティ (各 AWS アカウント ルートユーザーなど) に対する許可が制限されます。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの「SCP の仕組み」を参照してください。

- セッションポリシー - セッションポリシーは、ロールまたはフェデレーティッドユーザーの一時的なセッションをプログラムで作成する際に、パラメータとして渡す高度なポリシーです。結果としてセッションの権限の範囲は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」をご参照ください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、『IAM ユーザーガイド』の「ポリシーの評価ロジック」を参照してください。

アクセスコントロール

Amazon SNS には、AWS Identity and Access Management (IAM) ポリシーに使用されているのと同じ言語で記述されたポリシーを使用する独自のリソースベースのアクセス権限システムがあります。これは、Amazon SNS ポリシーや IAM ポリシーで同様の結果を達成できることを意味します。

> **Note**
>
> すべての AWS アカウント はアカウント内でユーザーにアクセス権限を委任できることを理解することが重要です。クロスアカウントアクセスによって、追加のユーザーを管理しなくても、AWSリソースへのアクセスを共有することができます。クロスアカウントのアクセスの詳細については、IAM ユーザーガイドの「クロスアカウントアクセスの有効化」を参照してください。

Amazon SNS でのアクセス管理の概要

このセクションでは、アクセスポリシー言語 を使用してポリシーを書き込むにあたって理解しておくべき基本的なコンセプトを説明します。また アクセスポリシー言語 と連携したアクセスコントロール法の一般的なプロセスやポリシーの評価方法も合わせて説明します。

トピック

アクセスコントロールを使用する場合

リソースへのアクセス権限やアクセス拒否の方法に高い柔軟性があります。ただし、一般的なユースケースは、次に示すように非常にシンプルなものとなっています。

- 他の AWS アカウント に対して、特定のトピックアクションタイプ (発行など) を許可します。詳細については、「トピックへの AWS アカウント アクセスの付与」を参照してください。
- HTTPS プロトコルに対してのみ、トピックのサブスクリプションを制限します。詳細については、「HTTPS へのサブスクリプションの制限」を参照してください。
- Amazon SNS が Amazon SQS キューにメッセージを発行することを許可する必要があります。詳細については、「Amazon SQS キューに発行する。」を参照してください。

主要なコンセプト

以下のセクションでは、アクセスポリシー言語を使用するにあたって理解しておくべきコンセプトを説明します。基本的なものから順に分かりやすく説明していきます。

トピック

- リクエスタ
- 評価
- エフェクト
- デフォルトで拒否
- 許可
- 明示的拒否

アクセス許可

アクセス権限とは、特定のリソースへのある種のアクセスに対し、許可または拒否をするというコンセプトです。アクセス権限は、基本的に「A は、条件 D に該当する C を対象とするアクション B の実行を許可または禁止されている」という形態をとります。例えば、ジェーン (A) は HTTP プロトコルを使用するのであれば(D) トピック A (C) へ発行することを (B) 許可されています。ジェーンがトピック A をパブリッシュした時点で、ジェーンにアクセス権限があるかどうか、またそのリクエストが定められているアクセス権限の条件を満たしているかどうかが、サービスによってチェックされます。

Statement

ステートメントとは、アクセスポリシー言語で使用するアクセス権限を定義する書式です。1 つのステートメントで 1 つのアクセス権限を定義します。ポリシー という、より広範囲のコンテナドキュメントの一部としてステートメントを書きます (次のコンセプト参照)。

ポリシー

ポリシーとは、1 つ以上のステートメントのコンテナの役目を果たすドキュメント (アクセスポリシー言語で記述) です。例えば、1 つのポリシーには「ジェーンは E メールプロトコルを使用してサブスクライブできる」というステートメントと、「ボブはトピック A を発行することができない」というステートメント、計 2 つのステートメントが含まれることがあります。以下の図に示されているように、同等のシナリオでは、「ジェーンは E メールプロトコルを使用してサブスクライブできる」というポリシーと、「ボブはトピック A を発行することができない」というポリシー、計 2 つのポリシーを含むことができます。

ポリシードキュメントでは ASCII 文字のみを使用できます。aws:SourceAccount と aws:SourceOwner を活用することで、非 ASCII 文字を含む他の AWS のサービスでの ARN をプラグインする必要があるシナリオを回避することができます。aws:SourceAccount と aws:SourceOwner の違いを確認してください。

Issuer

発行者とは、リソース用のアクセス権限についてのポリシーを記述する個人です。発行者は当然のこととして、リソースの所有者になります。AWS では AWS サービスユーザーは所有していないリソースのポリシーを作成できません。ジョンがリソース所有者である場合、AWS は、ジョンがそのリソースに関する許可を付与するために独自に記述したポリシーを送信するときに、ジョンのアイデンティティを認証します。

Principal

プリンシパルとは、ポリシーのアクセス権限を適用される個人またはグループを指します。「A は、条件 D に該当する C を対象とするアクション B の実行を許可または禁止されている」というステートメントにおいては、A がプリンシパルに相当します。ポリシーでは、「誰でも」プリンシパルに設定することができます (例えば、ワイルドカードを指定して全員に設定できます)。例えば、リクエスタの実際のアイデンティティをベースにアクセス制限をかけたくなければ、リクエスタの IP アドレスのような、他のアイデンティティの特性をもとにして行うことができます。

[アクション]

アクションとは、プリンシパルに対し、実行が許可されているアクティビティです。「A は、条件 D に該当する場合 C に対して B を実行することを許可されている」というステートメントにおいては、B がアクションに相当します。通常、アクションとは、リクエストに埋め込まれて AWS に渡されるオペレーションのことです。例えば、ジェーンが Amazon SNS に Action =Subscribe とい

うリクエストを送信します。1 つのポリシーに 1 つまたは複数のアクションを指定することができます。

リソース

リソースとは、プリンシパルがアクセスを要求するオブジェクトのことです。「A は、条件 D に該当する場合 C に対して B を実行することを許可されている」というステートメントにおいては、C がリソースに相当します。

条件とキー

条件とは、アクセス権限についての制限や詳細のことです。「A は、条件 D に該当する場合 C に対して B を実行することを許可されている」というステートメントにおいては、D が条件に相当します。ポリシーの中でも、記述が最も詳細かつ複雑になるのが、この条件部分です。よく使用される条件の設定項目は以下のとおりです。

- 日時 (特定の日付以前に到着したリクエストのみ処理するなど)
- IP アドレス (特定の CIDR 範囲内の IP アドレスからのリクエストのみ処理するなど)

キーは、アクセス制限に使用される基本項目です。例えば、リクエストの日時がこれに相当します。

制限は、条件とキーの両方を使用して定義します。具体例を挙げて説明します。2010 年 5 月 30 日以前のアクセスを制限するには、`DateLessThan` という条件を使用します。`aws:CurrentTime` と呼ばれるキーを使用して それを `2010-05-30T00:00:00Z` の値に設定します。AWS では使用できる条件やキーを定義します。AWS のサービス自体 (Amazon SQS や Amazon SNS など) によりサービス固有のキーが定義されている場合もあります。詳細については、「Amazon SNS API のアクセス許可: アクションとリソースのリファレンス」を参照してください。

リクエスタ

リクエスタとは、AWS のサービスにリクエストを送信する人、または特定のリソースへのアクセスを要求する人です。リクエスタが AWS に送信するリクエストの内容は、基本的には「条件 D に該当する場合 C に対して B を実行することを許可してください」のようになります。

評価

評価とは、AWS のサービスが受信したリクエストを拒否または許可するかを、該当するポリシーに基づいて判断するプロセスのことです。評価論理の詳細については、「評価論理」を参照してください。

エフェクト

エフェクトとは、ポリシーのステートメントが評価時に返す結果のことです。この値はポリシーのステートメントを記述するときに指定します。使用可能な値は deny と allow です。

例えば、南極大陸からのすべてのリクエストを拒否するステートメントを含むポリシーを記述できます (リクエストの送信元 IP アドレスが南極大陸に割り当てられている場合、エフェクトの値を deny とします)。または、南極大陸以外からのすべてのリクエストを許可するステートメントを含むポリシーを記述できます (リクエストの送信元が南極大陸でない場合、エフェクトの値を allow とします)。2 つのステートメントは同じことを行うように見えますが、アクセスポリシー言語の論理上では異なるものです。詳細については、「評価論理」を参照してください。

エフェクトに特定できる値は allow と deny の 2 つだけですが、ポリシーの評価結果には、デフォルトで拒否、許可および明示的拒否の 3 種類があります。詳細については、以下のコンセプトおよび「評価論理」を参照してください。

デフォルトで拒否

デフォルトで拒否とは、ポリシーに許可または明示的拒否が指定されていない場合に、デフォルトで適用される拒否のことです。

許可

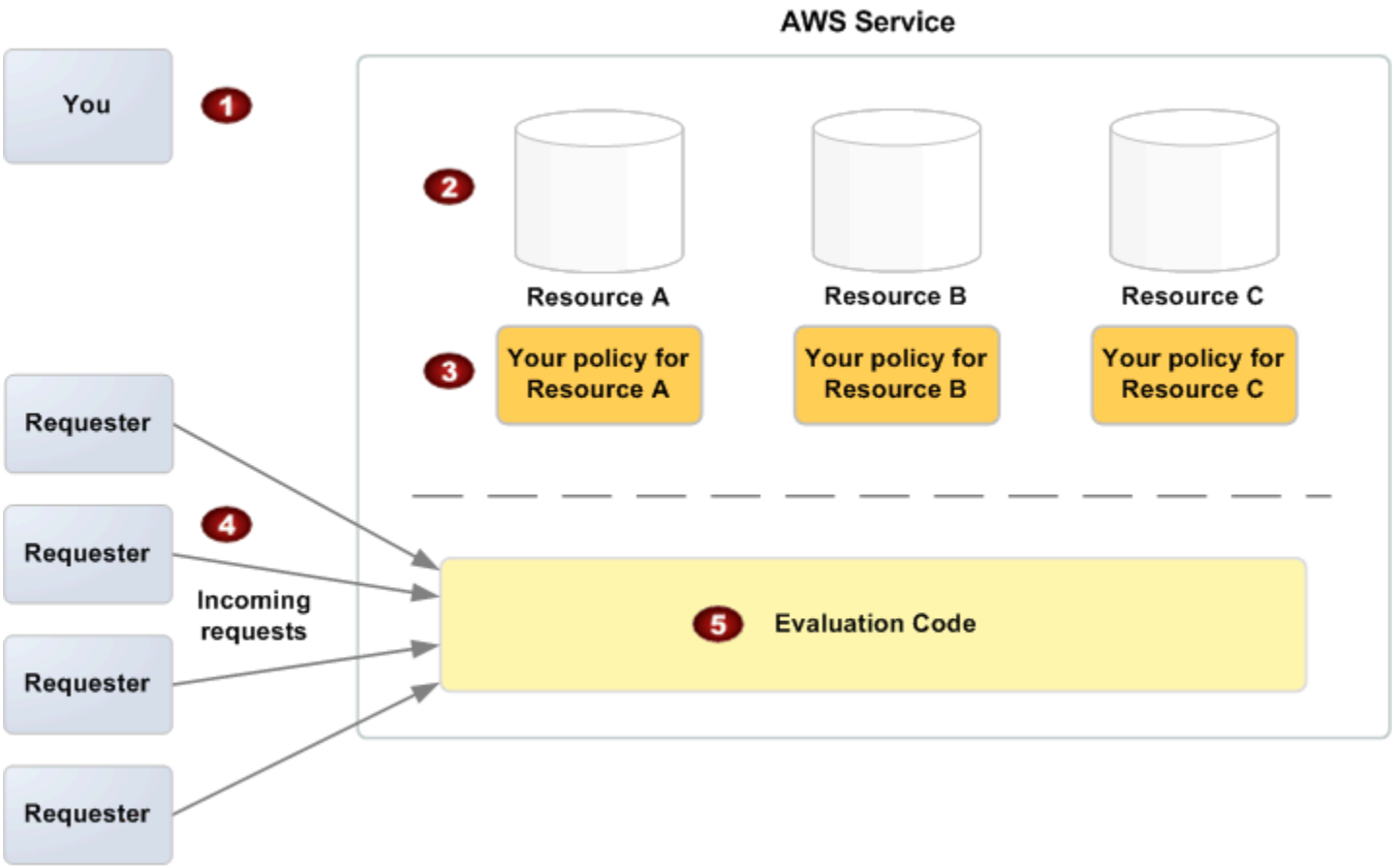
許可とは、ステートメントに effect=allow が指定されていて、許可条件がすべて満たされている場合に返される結果です。例えば、2010 年 4 月 30 日午後 1 時までに受信されたリクエストが許可されます。許可は、すべてのデフォルトで拒否に優先して適用されますが、明示的拒否が 1 つでもあれば適用されません。

明示的拒否

明示的拒否とは、ステートメントに effect=deny が指定されていて、拒否条件がすべて満たされている場合に返される結果です。例えば、送信元が南極大陸であるすべてのリクエストが拒否されます。その他のポリシーによって許可されている場合においても、南極から来たリクエストに対しては常に拒否します。

アーキテクチャの概要

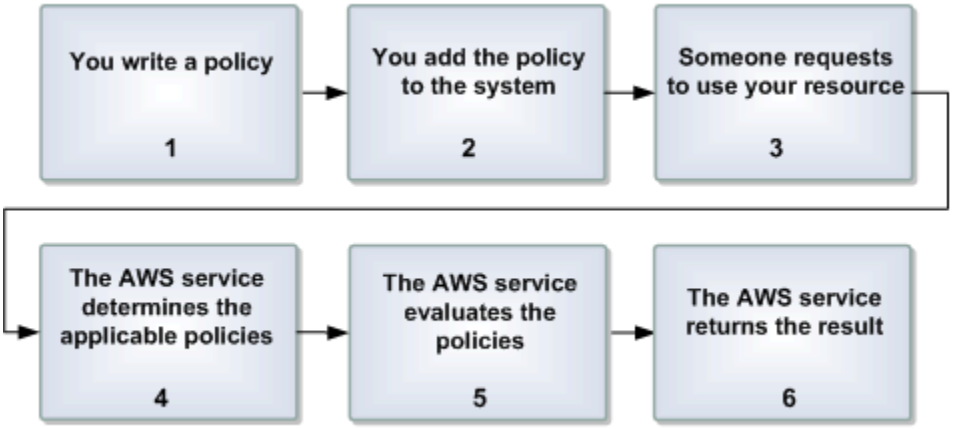
以下の図と表に、リソースのアクセスコントロールに関与する主要コンポーネントとそのインタラクティブな関わり合いを表します。



1	お客様 = リソース所有者。
2	リソース (AWS のサービスに含まれているもの、Amazon SQS キューなど)。
3	ポリシー。 通常、1 つのリソースに 1 つのポリシーを適用しますが、複数のポリシーを適用することも可能です。AWS サービスには、ポリシーのアップロードや管理に使用できる API が用意されています。
4	リクエスタ、および AWS のサービスに対するリクエスト。
5	アクセスポリシー言語の評価コード。 AWS サービス内のコードセットです。受信したリクエストと該当するポリシーを照合して、リクエスタにリソースへのアクセスを許可するかどうかを判定します。このサービスによる判定の詳細については、「評価論理」を参照してください。

アクセスポリシー言語を使用する

以下の図と表は、アクセスコントロールとアクセスポリシー言語の連携方法の通常プロセスを表しています。



アクセスコントロールをアクセスポリシー言語と使用するプロセス

1	リソース用ポリシーを記述します。 例えば、Amazon SNS トピックのアクセス権限を指定するポリシーを記述します。
2	AWS にポリシーをアップロードします。 AWS サービスでは、ポリシーのアップロードに使用できる API を提供しています。例えば、特定の Amazon SNS トピック用のポリシーをアップロードするために Amazon SNS `SetTopicAttributes` アクションを使用します。
3	ある人物から、リソースの使用許可を求めるリクエストが送信されます。 例えば、ユーザーがトピックの 1 つを利用するために Amazon SNS へリクエストを送信します。
4	どのポリシーがリクエストに適用可能であるか、AWS サービスによって決定されます。 例えば、Amazon SNS がすべての利用可能な Amazon SNS ポリシーを調べ、リソースの内容とリクエスタに基づいて、どのポリシーが適用可能であるかを決定します。
5	AWS サービスがポリシーを評価します。

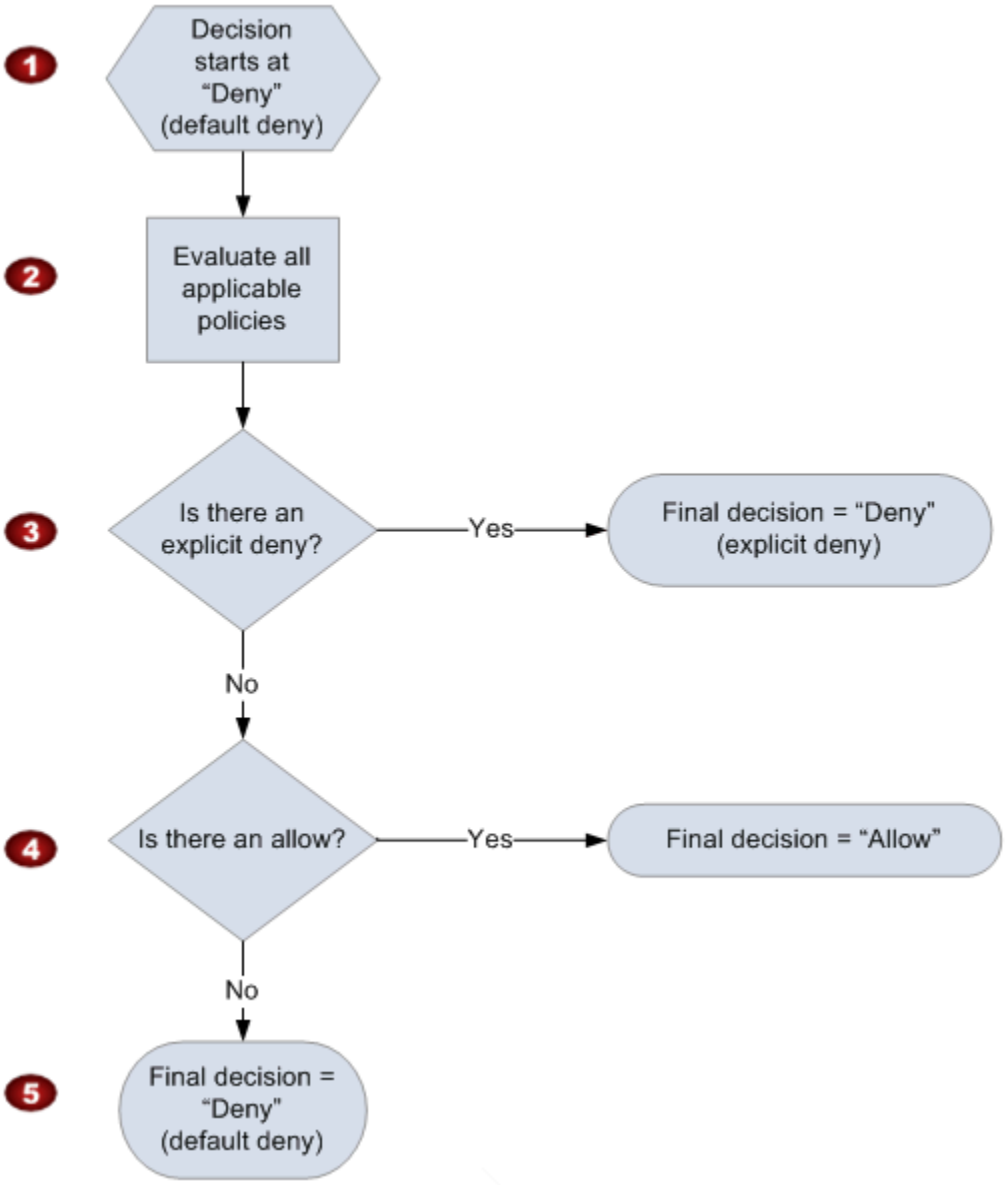
	例えば、Amazon SNS は、ポリシーを評価し、トピックの使用許可をリクエスタに付与するかどうかを決定します。決定論理の詳細については、「評価論理」を参照してください。
6	AWS サービスにおいては、リクエスタを拒否するか、またはプロセスを継続するかのどちらかが行われます。 例えば、ポリシーの評価結果に基づいて、サービスによって「アクセス拒否」エラーがリクエスタに返されるか、リクエストのプロセスを継続するかのどちらかが行われます。

評価論理

評価時の目標は、特定のリクエスト付与を許可するか拒否するかを判断することです。評価論理は、以下の複数の基本ルールに従っています。

- デフォルトでは、リソースの使用許可を求めるリクエストについては、リクエスタが自分自身である場合を除いて、拒否を適用する
- 許可はすべてのデフォルトで拒否に優先する
- 明示的拒否はすべての許可に優先する
- ポリシー評価の順序は重要ではない

以下のフローチャートと考察では、決定方法についての詳細説明を紹介します。



1	決定はデフォルトで拒否から始まります。
2	次に、エンフォースメントコードは、リクエストに適用可能なポリシーすべてを、リソース、プリンシパル、アクション、および条件に基づいて評価します。 エンフォースメントコードによるポリシー評価の順序は重要ではありません。
3	前述のすべてのポリシーにおいて、リクエストに適応する明示的拒否のインストラクションがエンフォースメントコードによって検索されます。

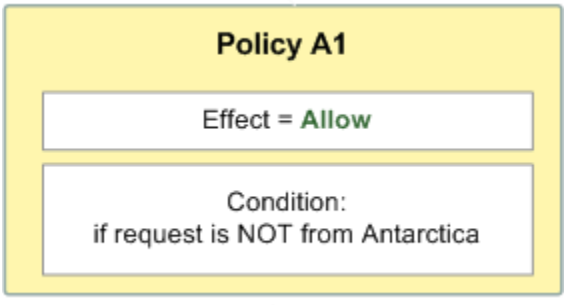
	仮に 1 つでも見つかった場合、エンフォースメントコードは「拒否」の決定を返し、プロセスを終了します (これは明示的拒否となります。詳細については、「明示的拒否」を参照してください)。
4	明示的拒否が見つからなかった場合、リクエストに適応する「許可」のインストラクションがエンフォースメントコードによって検索されます。 仮に 1 つでも見つかった場合、エンフォースメントコードは「許可」の決定を返し、プロセスは完了します (サービスはリクエストのプロセスを継続します)。
5	許可が見つからなかった場合、最終決定は「拒否」となります (明示的拒否または許可が見つからない場合、デフォルトで拒否として見なされるためです (詳細については、「デフォルトで拒否」を参照してください)。

明示的拒否とデフォルトで拒否の相互作用

ポリシーがリクエストに直接適用されない場合の結果は、デフォルトで拒否となります。例えば、ユーザーが Amazon SNS の使用をリクエストするが、トピックのポリシーではユーザーの AWS アカウント を全く参照しない場合、そのポリシーの適用結果はデフォルトで拒否となります。

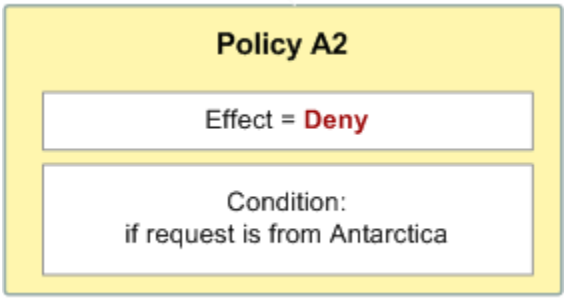
ステートメントの条件が満たされていない場合においても、ポリシーの結果としてデフォルトで拒否となります。ステートメントのすべての条件が満たされている場合、ポリシーのエフェクトエレメントの値に基づいて、ポリシーの結果は許可または明示的拒否のどちらかとなります。条件が満たされていない際にポリシーが行為を特定していない場合、デフォルトの結果としてデフォルトで拒否となります。

例えば、南極大陸から来るリクエストを防ぐとします。その場合、南極大陸から来ていないリクエストにのみ許可を与えるポリシー (ポリシー A1 とする) を記述します。以下の図はポリシーについて解説しています。



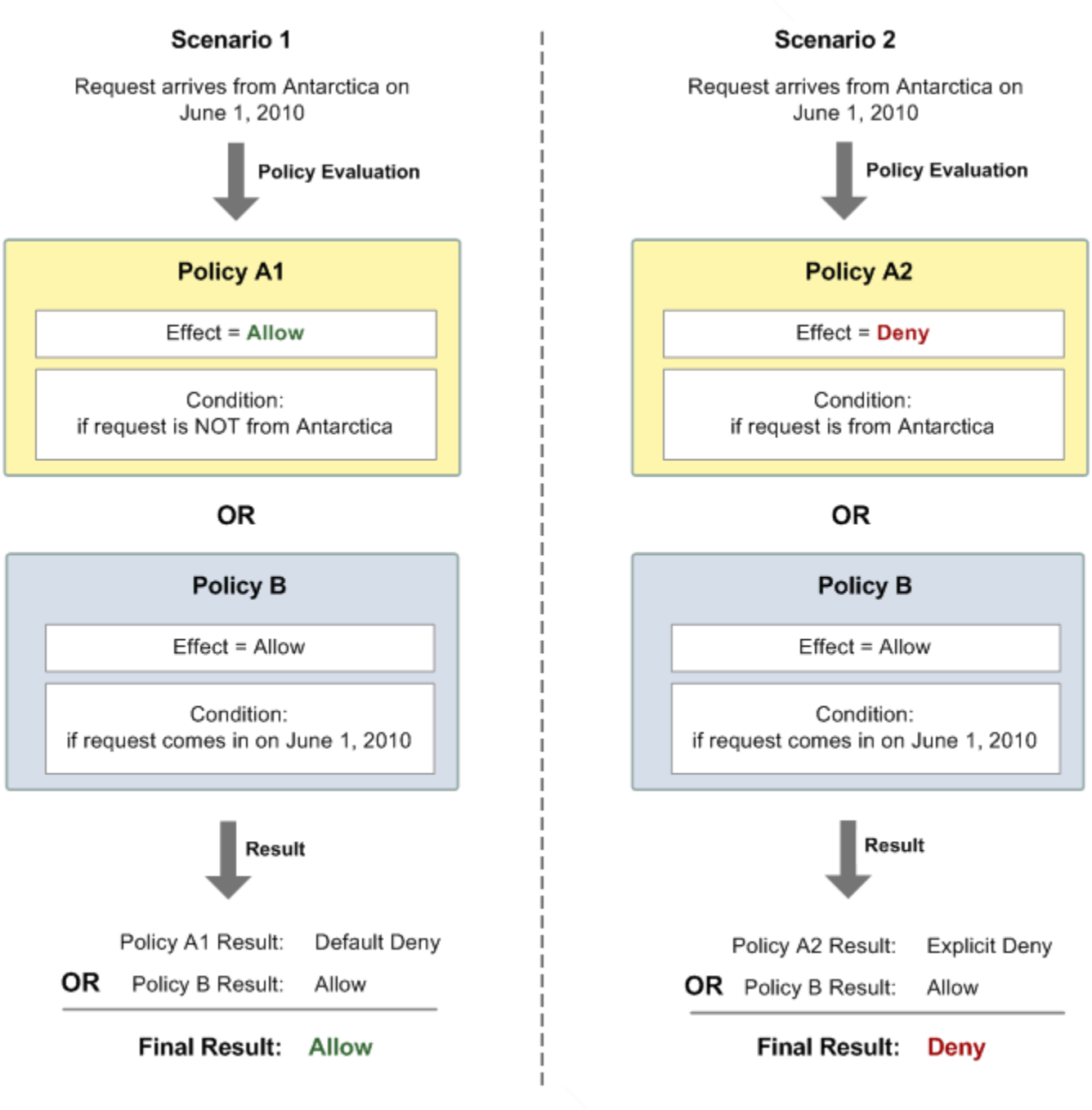
リクエストがアメリカから送られてきた場合、条件を満たしています (リクエストが南極大陸からのものでないため)。従って、そのリクエストは許可されます。しかし、リクエストが南極大陸から送られてきた場合、条件を満たしていないため、ポリシーの結果としてデフォルトで拒否となります。

以下の図のとおり、ポリシー (ポリシー A2 とする) を書き換えることにより、結果を明示的拒否に変えることができます。南極大陸から送られてきた場合、ポリシーによってリクエストが明示的に拒否されます。



リクエストが南極大陸から送られてきた場合、条件を満たしているため、ポリシーの結果として明示的な拒否となります。

デフォルトで拒否と明示的拒否の区別は重要です。許可によってデフォルトで拒否は上書きできますが、明示的拒否は上書きできないためです。例えば、リクエストが 2010 年 6 月 1 日に到着すれば許可するという別のポリシーがあるとしましょう。このポリシーが、南極大陸からのアクセスを制限しているポリシーと併用されている場合、全体の結果にどのような影響を及ぼすでしょうか? 日付ベースのポリシー (ポリシー B とする) が前述のポリシー A1 および A2 と併用されている場合、全体の結果が比較されます。シナリオ 1 は、ポリシー A1 とポリシー B が併用されている場合、シナリオ 2 は、ポリシー A2 とポリシー B が併用されている場合です。以下の図と考察は、2010 年 6 月 1 日に南極大陸からリクエストが来た場合についての結果を示しています。



このセクションの最初に説明したとおり、シナリオ 1 においては、ポリシー A1 はデフォルトで拒否を返します。2010 年 6 月 1 日に到着したリクエストは、当然のことながら許可されるため、ポリシー B は許可を返します。ポリシー B による許可は、ポリシー A1 のデフォルトで拒否に優先するため、結果としてリクエストは許可されます。

このセクションの最初に説明したとおり、シナリオ 2 においては、ポリシー A2 は明示的拒否を返します。再度、ポリシー B は許可を返します。ポリシー A2 による明示的拒否は、ポリシー B の許可に優先するため、結果としてリクエストは拒否されます。

Amazon SNS アクセスコントロールのケース例

このセクションでは、アクセスコントロールの一般的なユースケース例をいくつか紹介します。

トピック

トピックへの AWS アカウント アクセスの付与

Amazon SNS システムにトピックがあるとします。最も簡単なケースとして、特定のトピックアクション (発行など) へのアクセスを 1 つ以上の AWS アカウント に許可するとします。

これは、Amazon SNS API アクション `AddPermission` を使用して実行できます。これには、トピック、AWS アカウント ID リスト、アクションリスト、ラベルが必要ですが、トピックのアクセスコントロールポリシー内に新規ステートメントが自動的に作成されます。この場合、 Amazon SNS により新規ポリシーステートメントが自動的に作成されます。手動でポリシーを記述する必要はありません。ラベルと共に `RemovePermission` を呼び出すことによって、後日ポリシーステートメントを削除することができます。

例えば、トピック arn:aws:sns:us-east-2:444455556666:MyTopic、AWS アカウントID 1111-2222-3333、 `Publish`アクション、ラベル `AddPermission`で を呼び出した場合`grant-1234-publish`、Amazon SNS は次のアクセスコントロールポリシーステートメントを生成して挿入します。

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
```

```
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

一度このステートメントが追加されてしまえば、AWS アカウント 1111-2222-3333 のユーザーはトピックへメッセージを発行することができます。

HTTPS へのサブスクリプションの制限

以下の例では、通知配信プロトコルを HTTPS に制限します。

Amazon SNS AddPermission アクションでは、トピックへのアクセスを許可できるだけです。プロトコル制限を指定するには、独自のポリシーを記述する必要があります。つまり、独自のポリシーを記述し、Policy アクションを使用してトピックの SetTopicAttributes 属性をその新しいポリシーに設定します。

以下の例では、全ポリシーが、AWS アカウント ID 1111-2222-3333 にトピックからの通知サブスクリプション能力を付与しています。

```
{
  "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"
      }
    }
  }]
}
```

Amazon SQS キューに発行する。

このユースケースでは、トピックから Amazon SQS キューへのメッセージを発行するものとします。Amazon SNS のように、Amazon SQS も Amazon のアクセスコントロールポリシー言語を使用

します。Amazon SNS からのメッセージ送信を許可するには、キューにおいてポリシーを設定するために Amazon SQS アクション `SetQueueAttributes` を使用することが必要となります。

Amazon SQS `AddPermission` アクションでは、条件付きポリシーステートメントを作成できません。これを作成するには、独自のポリシーを記述する必要があります。

> **Note**
>
> 以下の例は、Amazon SQS ポリシー (キューへのアクセスコントロール) を示すもので、Amazon SNS ポリシー (トピックへのアクセスコントロール) を示すものではありません。アクションとは Amazon SQS アクションのことで、リソースとはキューの Amazon リソースネーム (ARN) のことです。`GetQueueAttributes` アクションと共にキューの `QueueArn` 属性を取得することにより、キューの ARN を決定する事ができます。

```
{
  "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"
      }
    }
  }]
}
```

このポリシーには、キューへ送信されるメッセージのソースに基づいたキューへのアクセス制限のための `aws:SourceArn` 条件が使用されます。お客様ご自身のトピックの 1 つから送信されたメッセージであった場合、 Amazon SNS に対し、キューからのメッセージ送信を許可するために、このタイプのポリシーを使用することができます。この場合、ARN が arn:aws:sns:us-east-2:444455556666: である特定のトピックを指定しますMyTopic。

前述のポリシーを例として、Amazon SQS ポリシーを記述したり、特定のキューに追加することができます。Amazon SNS およびその 他 AWS のサービスへのアクセスを許可します。Amazon SNS

は、新しく作成されたすべてのトピックにデフォルトポリシーを付与します。デフォルトのポリシーでは、他のすべての AWS サービスに対するトピックへのアクセスを付与します。このデフォルトポリシーは aws:SourceArn 条件を使用して、お客様が所有する AWS リソースに代わる場合にのみ AWS サービスがお客様のトピックにアクセスすることを確実にします。

Amazon S3 イベント通知がトピックに発行することを許可する

この場合、別の AWS アカウント の Amazon S3 バケットがお客様のトピックへの発行ができるよう、トピックのポリシーを設定します。Amazon S3 からの通知の発行の詳細については、「バケットイベントの通知の設定」を参照してください。

この例では、お客様が自身でポリシーの記述をするものと仮定され、お客様の新規ポリシーへのトピックの Policy 属性を設定するための SetTopicAttributes アクションとして使用されます。

以下のステートメント例では、SourceAccount 条件を使用して、Amazon S3 所有者アカウントのみがトピックにアクセスできるようにしています。この例では、トピック所有者は 111122223333、また Amazon S3 所有者は 444455556666 となっています。この例では、444455556666 が所有するすべての Amazon S3 バケットが への発行を許可されていることが示されています MyTopic。

```
{
  "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"
      }
    }
  }]
}
```

Amazon SNS にイベントを発行する場合、以下のサービスで aws:SourceAccount を使用できます。

- Amazon API Gateway
- Amazon CloudWatch

- Amazon DevOpsGuru
- Amazon ElastiCache
- Amazon GameLift
- Amazon Pinpoint SMS および音声 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

Amazon SES が別のアカウントが所有するトピックへの発行を許可する

AWS のサービスに別の AWS アカウント によって所有されているトピックに発行する許可を付与することができます。111122223333 アカウントにサインインし、Amazon SES を開き、E メールを作成したとします。この E メールに関する通知を、444455556666 アカウントが所有する Amazon SNS トピックに発行するには、以下のようなポリシーを作成します。そのためには、プリンシパル (他のサービス) と各リソースの所有権に関する情報を提供する必要があります。`Resource` ステートメントは、トピック所有者のアカウント ID 444455556666 を含むトピック ARN を提供します。`"aws:SourceOwner": "111122223333"` ステートメントは、アカウントがその E メールを所有することを指定します。

```
{
  "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"
        }
      }
    }
  ]
}
```

Amazon SNS にイベントを発行する場合、以下のサービスで aws:SourceOwner を使用できます。

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOpsGuru
- Amazon ElastiCache
- Amazon GameLift
- Amazon Pinpoint SMS および音声 API
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

aws:SourceAccount と aws:SourceOwner

> **Important**
>
> aws:SourceOwner は廃止されました。新しいサービスを Amazon SNS と統合できるのは、aws:SourceArn と aws:SourceAccount を使用した場合のみです。Amazon SNS は、現在 aws:SourceOwner をサポートしている既存のサービスとの下位互換性を引き続き維持しています。

aws:SourceAccount 条件キーと aws:SourceOwner 条件キーは、一部の AWS のサービスが Amazon SNS トピックに発行するときに、それぞれ設定されます。サポートされている場合、値は 12 桁の AWS アカウント ID で、サービスがアカウントに代わってデータを発行します。サービスによってサポートするものは変わります。

- Amazon S3 通知がどのように aws:SourceAccount を使用するか、およびその条件をサポートする AWS サービスのリストに関しては「Amazon S3 イベント通知がトピックに発行することを許可する」を参照してください。
- Amazon SES 通知がどのように aws:SourceOwner を使用するか、およびその条件をサポートする AWS サービスのリストに関しては「Amazon SES が別のアカウントが所有するトピックへの発行を許可する」を参照してください。

AWS Organizations の組織のアカウントが別のアカウントのトピックに発行することを許可する

AWS Organizations サービスは、請求の一元管理、アクセスとセキュリティの制御、AWS アカウント 間でのリソースの共有に役立ちます。

組織 ID は Organizations コンソールで確認できます。詳細については、「管理アカウントからの組織の詳細の表示」を参照してください。

この例では、組織 myOrgId の任意の AWS アカウント が、アカウント 444455556666 の Amazon SNS トピック MyTopic に発行できます。ポリシーでは、aws:PrincipalOrgID グローバル条件キーを使用して組織 ID 値を確認します。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
```

```
}
```

CloudWatch アラームが別のアカウントのトピックに発行することを許可する

この場合、アカウント の CloudWatch アラーム111122223333は、アカウント の Amazon SNS トピックに発行できます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:*"
                }
            }
        }
    ]
}
```

Amazon SNS トピックの発行を特定の VPC エンドポイントのみからに制限する

この場合、アカウント 444455556666 内のトピックは、ID vpce-1ab2c34d の VPC エンドポイントのみからの発行を許可されます。

```
{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "SNS:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
```

```
  }]
}
```

Amazon Simple Notification Service で IAM を使用する方法

IAM を使用して Amazon SNS へのアクセスを管理する前に、Amazon SNS で使用できる IAM 機能について理解しておく必要があります。

Amazon Simple Notification Service で使用できる IAM の機能

IAM 機能	Amazon SNS のサポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	はい
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	なし
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	あり
プリンシパル権限	あり
サービスロール	あり
サービスリンクロール	いいえ

Amazon SNS および AWS のその他のサービスが大部分の IAM 機能とどのように動作するかに関するおおまかな説明については、IAM ユーザーガイドの「IAM と連携する AWS のサービス」を参照してください。

Amazon SNS のポリシーアクション

ポリシーアクションに対するサポート	あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーのAction要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連する AWS API オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

Amazon SNS アクションの一覧については、サービス認証リファレンスの「Resources Defined by Amazon Simple Notification Service」(Amazon Simple Notification Service で定義されるリソース) を参照してください。

Amazon SNS のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
sns
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

Amazon SNS のアイデンティティベースポリシーの例を確認するには、「Amazon Simple Notification Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon SNS のポリシーリソース

ポリシーリソースに対するサポート	あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

`Resource` JSON ポリシーの要素は、オブジェクトあるいはアクションが適用されるオブジェクトを指定します。ステートメントには、`Resource` または `NotResource` 要素を含める必要があります。ベストプラクティスとしては、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon SNS リソースタイプとその ARN の一覧については、サービス認証リファレンスの「Actions Defined by Amazon Simple Notification Service」 (Amazon Simple Notification Service で定義されるアクション) を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Resources Defined by Amazon Simple Notification Service」(Amazon Simple Notification Service で定義されるリソース) を参照してください。

Amazon SNS のアイデンティティベースポリシーの例を確認するには、「Amazon Simple Notification Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon SNS のポリシー条件キー

サービス固有のポリシー条件キーのサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。equal や less than などの条件演算子を使用して条件式を作成することによって、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素が指定されている場合、または 1 つの Condition 要素に複数のキーが指定されている場合、AWS では AND 論理演算子を使用してそれらを評価します。単一の条件キーに複数の値が指定されている場合、AWS では OR 論理演算子を使用して条件を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる許可を付与できます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS はグローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、『IAM ユーザーガイド』の「AWS グローバル条件コンテキストキー」を参照してください。

Amazon SNS 条件キーの一覧については、サービス認証リファレンスの「Condition Keys for Amazon Simple Notification Service」(Amazon Simple Notification Service の条件キー) を参照してください。条件キーを使用できるアクションとリソースについては、「Resources Defined by Amazon Simple Notification Service」(Amazon Simple Notification Service で定義されるリソース) を参照してください。

Amazon SNS のアイデンティティベースポリシーの例を確認するには、「Amazon Simple Notification Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon SNS での ACL

ACL のサポート	なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon SNS での ABAC

ABAC (ポリシー内のタグ) のサポート	部分的

属性ベースのアクセスコントロール (ABAC) は、属性に基づいて権限を定義する認可戦略です。AWS では、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール)、および多数の AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。次に、プリンシパルのタグがアクセスを試行するリソースのタグと一致したときにオペレーションを許可するよう、ABAC ポリシーを設計します。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを制御するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素 でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値は Yes です。サービスが一部のリソースタイプに対してのみ 3 つの条件キーすべてをサポートする場合、値は Partial です。

ABAC の詳細については、『IAM ユーザーガイド』の「ABAC とは?」を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、「IAM ユーザーガイド」の「属性に基づくアクセスコントロール (ABAC) を使用する」を参照してください。

Amazon SNS での一時的な認証情報の使用

一時的な認証情報のサポート	あり

AWS のサービス には、一時的な認証情報を使用してサインインしても機能しないものがあります。一時的な認証情報で機能する AWS のサービスなどの詳細については、「IAM ユーザーガイド」の「IAM と連携する AWS のサービス」を参照してください。

ユーザー名とパスワード以外の方法で AWS Management Console にサインインする場合は、一時的な認証情報を使用していることになります。例えば、会社のシングルサインオン (SSO) リンクを使用して AWS にアクセスすると、そのプロセスは自動的に一時的な認証情報を作成します。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ロールへの切り替え (コンソール)」を参照してください。

一時的な認証情報は、AWS CLI または AWS API を使用して手動で作成できます。作成後、一時的な認証情報を使用して AWS にアクセスできるようになります。AWS は、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することをお勧めします。詳細については、「IAM の一時的なセキュリティ認証情報」を参照してください。

Amazon SNS のクロスサービスプリンシパル許可

フォワードアクセスセッション (FAS) をサポート	はい

IAM ユーザーまたはロールを使用して AWSでアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行してから、別のサービスの別のアクションを開始することがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Amazon SNS のサービスロール

サービスロールに対するサポート	あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、『IAM ユーザーガイドの「AWS のサービス に権限を委任するロールの作成」を参照してください。

> **Warning**
>
> サービスロールの許可を変更すると、Amazon SNS の機能が破損する可能性があります。Amazon SNS が指示する場合以外は、サービスロールを編集しないでください。

Amazon SNS のサービスにリンクされたロール

サービスにリンクされたロールのサポート	いいえ

サービスにリンクされたロールは、AWS のサービスにリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの許可を表示できますが、編集はできません。

サービスにリンクされたロールの作成または管理の詳細については、「IAM と提携する AWS のサービス」を参照してください。表の中から、[サービスにリンクされたロール] (サービスにリンクされたロール) 列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon Simple Notification Service のアイデンティティベースのポリシーの例

デフォルトでは、ユーザーおよびロールには Amazon SNS リソースを作成または変更する許可はありません。また、AWS Management Console、AWS Command Line InterfaceAWS CLI、または AWS API を使用してタスクを実行することもできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者がロールに IAM ポリシーを追加すると、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

Amazon FSx が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、サービス認証リファレンスの「Actions, Resources, and Condition Keys for Amazon Simple Notification Service」(Amazon Simple Notification Service のアクション、リソース、および条件キー) を参照してください。

トピック

- ポリシーのベストプラクティス
- Amazon SNS コンソールの使用
- 他のポリシータイプ
- 複数のポリシータイプ
- ユーザーが自分のアクセス許可を表示できるようにする方法

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内で誰かが Amazon SNS リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、AWS アカウント に料金が発生する可能性があります。アイデンティティベースのポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください。

- AWS マネージドポリシーを使用して開始し、最小特権の権限に移行する – ユーザーとワークロードへの権限の付与を開始するには、多くの一般的なユースケースのために権限を付与する AWS マネージドポリシーを使用します。これらは AWS アカウントで使用できます。ユースケースに応じた AWS カスタマー管理ポリシーを定義することで、許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する - IAM ポリシーで許可を設定するときは、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権権限とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーと権限」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。また、AWS のサービス など特定の AWS CloudFormation を介して使用する場合、条件を使用してサービスアクションへのアクセスを許可することもできます。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素: 条件」を参照してください。
- IAM アクセスアナライザーを使用して IAM ポリシーを検証し、安全で機能的な許可を確保する - IAM アクセスアナライザーは、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM アクセスアナライザーによるポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する - AWS アカウント で IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、『IAM ユーザーガイド』の「IAM でのセキュリティのベストプラクティス」を参照してください。

Amazon SNS コンソールの使用

Amazon Simple Notification Service コンソールにアクセスするには、許可の最小限のセットが必要です。アクセス許可により、AWS アカウント の Amazon SNS リソースの詳細をリストおよび表示できます。最小限必要なアクセス許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) ではコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソール権限を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスを許可します。

ユーザーとロールが引き続き Amazon SNS コンソールを使用できるようにするには、エンティティに Amazon SNS *ConsoleAccess* または *ReadOnly* AWS 管理ポリシーもアタッチします。詳細については、IAM ユーザーガイドの「ユーザーへの許可の追加」を参照してください。

他のポリシータイプ

AWS では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与される最大の許可を設定できます。

- 権限の境界 - 権限の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティに権限の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとその権限の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、権限の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。権限の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティの権限の境界」を参照してください。
- サービスコントロールポリシー (SCP) - SCP は、AWS Organizations で組織や組織単位 (OU) の最大許可を指定する JSON ポリシーです。AWS Organizations は、ユーザーのビジネスが所有する複数の AWS アカウント をグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP ではメンバーアカウントのエンティティ (各 AWS アカウント ルートユーザーなど) に対する許可が制限されます。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーティッドユーザーの一時的なセッションをプログラムで作成する際に、パラメータとして渡す高度なポリシーです。結果としてセッションの権限の範囲は、ユーザーまたはロールのアイデンティティベースポリシーとセッ

ションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」をご参照ください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、『IAM ユーザーガイド』の「ポリシーの評価ロジック」を参照してください。

ユーザーが自分のアクセス許可を表示できるようにする方法

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を、IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了する権限が含まれています。

```
{
    "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": "*"
        }
    ]
}
```

Amazon SNS のアイデンティティベースの ポリシー

アイデンティティベースポリシーをサポートする	あり

アイデンティティベースのポリシーは、IAM ユーザー、ユーザーグループ、ロールなどのアイデンティティにアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それがアタッチされているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素について学ぶには、『IAM ユーザーガイド』の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Amazon SNS のアイデンティティベースのポリシー例

Amazon SNS のアイデンティティベースポリシーの例を確認するには、「Amazon Simple Notification Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon SNS 内のリソースベースのポリシー

リソースベースのポリシーのサポート	はい

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーの例には、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる AWS アカウント にある場合、信頼できるアカウントの IAM 管理者は、リソースへのアクセス許可をプリンシパルエンティティ (ユーザーまたはロール) に付与する必要もあります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーを追加する必要はありません。詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

Amazon SNS でのアイデンティティベースのポリシーを使用する

トピック

- IAM ポリシーと Amazon SNS ポリシーを一緒に使う
- Amazon SNS リソース ARN 形式
- Amazon SNS API アクション
- Amazon SNS ポリシーキー
- Amazon SNS のポリシーの例

お客様の AWS アカウント のユーザーが Amazon SNS リソースで実行できる Amazon SNS アクションを指定できるよう、Amazon Simple Notification Service は AWS Identity and Access Management (IAM) と統合します。ポリシーで特定のトピックを指定できます。例えば、`Publish` アクションを AWS アカウント の特定のトピックで使用するアクセス権限を組織の特定のユーザーに付与する IAM ポリシーを作成するときに、変数を使用できます。詳細については、『IAM の使用』ガイドの「ポリシー変数」を参照してください。

> **Important**
>
> IAM で Amazon SNS を使用しても、Amazon SNS の使用方法は変わりません。Amazon SNS アクションに変更はなく、ユーザーおよびアクセスコントロールに関連する新しい Amazon SNS アクションはありません。

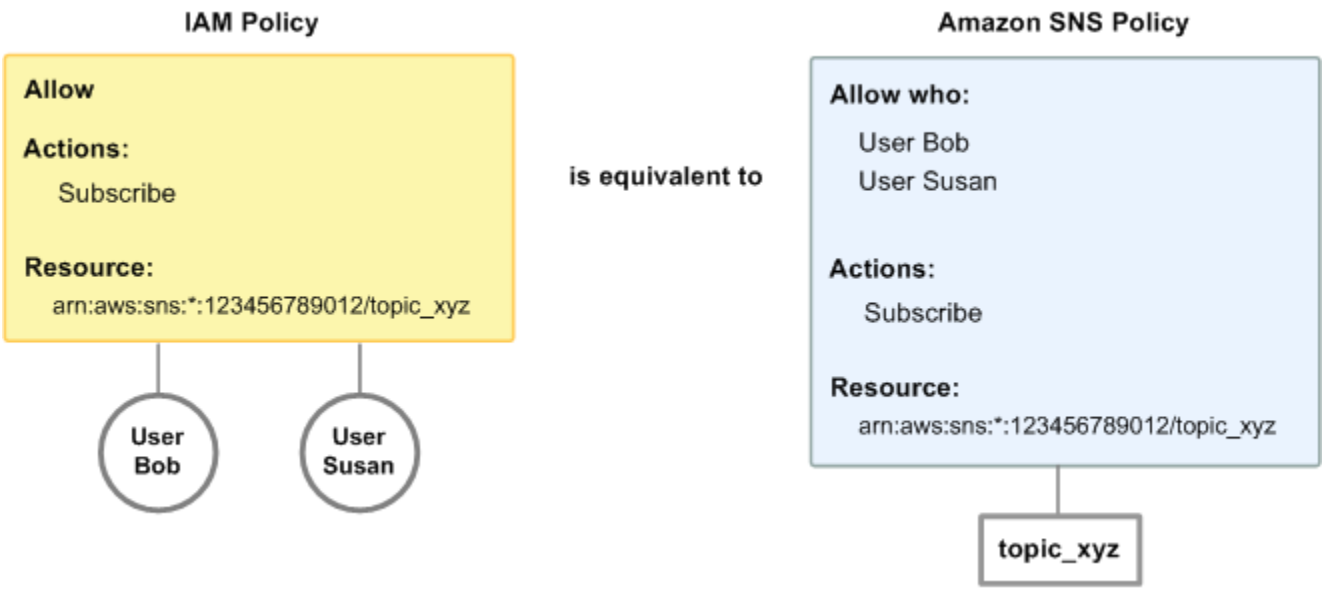
Amazon SNS アクションおよびリソースに対応するポリシーの例については、「Amazon SNS のポリシーの例」を参照してください。

IAM ポリシーと Amazon SNS ポリシーを一緒に使う

IAM ポリシーを使用して、Amazon SNS アクションおよびトピックへのユーザーのアクセスを制限します。IAM ポリシーでは、他の AWS アカウント ではなく、自分の AWS アカウント内のユーザーにのみに対してアクセスを制限できます。

特定のトピックに関連付ける Amazon SNS ポリシーにより、そのトピックを操作できるユーザー (トピックにメッセージを発行できるユーザー、サブスクライブできるユーザーなど) を制限できます。Amazon SNS ポリシーは、他の AWS アカウント、または自分の AWS アカウント 内のユーザーにアクセス権を付与できます。

Amazon SNS トピックのアクセス権限をユーザーに付与するには、IAM ポリシー、Amazon SNS ポリシー、またはその両方を使用できます。ほとんどの場合、どちらでも同じ結果が得られます。例えば、以下の図は、同じ働きを持つ IAM ポリシーと Amazon SNS ポリシーを示しています。IAM ポリシーでは、AWS アカウントの topic_xyz というトピックに対して Amazon SNS `Subscribe` アクションを許可します。IAM ポリシーは、ユーザーのボブとスーザンに添付されています (つまり、ボブとスーザンはポリシーに記述されているアクセス権限を持っています)。同様に、Amazon SNS ポリシーは topic_xyz の `Subscribe` にアクセスする許可をボブとスーザンに付与します。



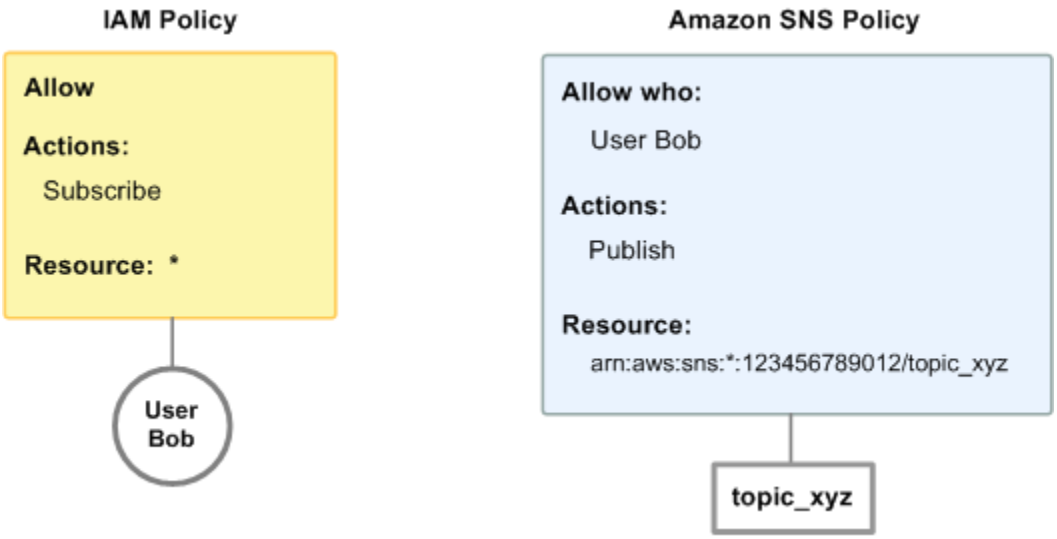
> **Note**
>
> 先の例では、条件のない単純なポリシーを示しました。どちらのポリシーでも特定の条件を指定して、同じ結果を得ることができます。

AWS IAM ポリシーと Amazon SNS ポリシーには、Amazon SNS ポリシーシステムでは他の AWS アカウント にアクセス許可を付与できますが、IAM ポリシーではできないという違いが 1 つあります。

両方のシステムを同時に使用してどのようにアクセス許可を管理するかは、ニーズに応じて決めてください。以下の例では、2 つのポリシーシステムがどのように連携するかを示しています。

Example 1

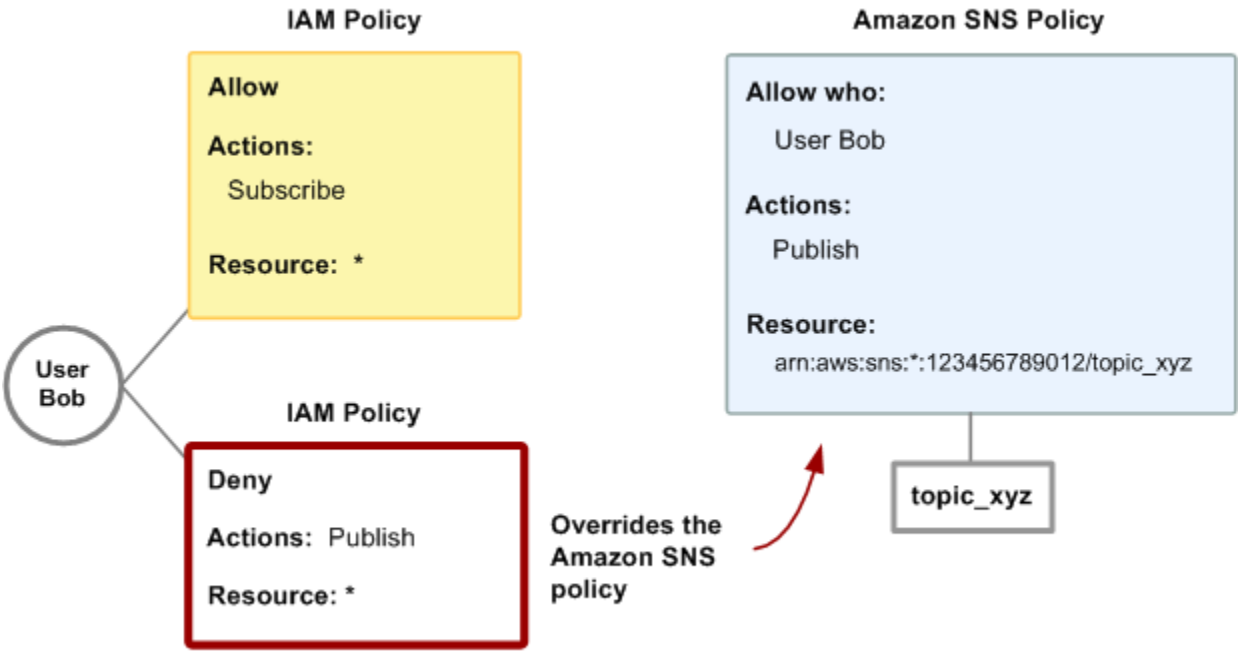
この例では、IAM ポリシーおよび Amazon SNS ポリシーの両方がボブに適用されます。IAM ポリシーは、任意の AWS アカウント のトピックに対してもボブに `Subscribe` へのアクセス権限を付与しますが、Amazon SNS ポリシーは特定のトピック (topic_xyz) において `Publish` を使用するアクセス権限をボブに付与します。以下の図に、そのコンセプトを示します。



ボブが AWS アカウントの任意のトピックをサブスクライブするリクエストを送信した場合、そのアクションは IAM ポリシーで許可されます。ボブが topic_xyz にメッセージを発行するリクエストを送信すると、そのアクションは Amazon SNS ポリシーで許可されます。

Example 2

この例は、例 1 で示した、ボブに 2 つのポリシーが適用されている状態に基づいています。ボブは、必要のない topic_xyz にメッセージを発行しているため、トピックに発行する機能を完全に削除するとします。最も簡単な方法は、すべてのトピックに対するボブの `Publish` アクションを拒否するような IAM ポリシーを追加することです。この 3 番目のポリシーは、topic_xyz に発行するアクセス権限を最初にボブに与えた Amazon SNS ポリシーより優先されます。明示的拒否は、常に許可よりも優先されるためです (ポリシー評価のロジックの詳細については、「評価論理」を参照)。以下の図に、そのコンセプトを示します。



Amazon SNS アクションおよびリソースに対応するポリシーの例については、「Amazon SNS のポリシーの例」を参照してください。Amazon SNS ポリシーの記述の詳細については、「Amazon SNS の技術文書」を参照してください。

Amazon SNS リソース ARN 形式

Amazon SNS では、トピックはポリシーで指定できる唯一のリソースタイプです。トピックの Amazon リソースネーム (ARN) 形式を以下に示します。

```
arn:aws:sns:region:account_ID:topic_name
```

ARN の詳細については、『IAM ユーザーガイド』の「ARN」を参照してください。

Example

以下は、123456789012 に属する米国東部 2 リージョン MyTopic の という名前のトピックの ARN AWS アカウント です。

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

Example

Amazon SNS がサポートする異なるリージョンのそれぞれ MyTopic に という名前のトピックがある場合は、次の ARN を使用してトピックを指定できます。

```
arn:aws:sns:*:123456789012:MyTopic
```

トピック名にはワイルドカード * および ? を使用できます。例えば、以下では、ボブが bob_ をプレフィックスとして付けて作成したすべてのトピックを参照します。

```
arn:aws:sns:*:123456789012:bob_*
```

利便性を高めるため、トピックを作成すると、Amazon SNS は応答でトピックの ARN を返します。

Amazon SNS API アクション

IAM ポリシーでは、Amazon SNS で提供されている任意のアクションを指定できます。ただし、`ConfirmSubscription` および `Unsubscribe` アクションでは認証を必要としません。つまり、ポリシーでこれらのアクションを指定しても、IAM はこれらのアクションへのユーザーのアクセスを制限しません。

ポリシーで指定する各アクションには、小文字の文字列 `sns:` をプレフィックスとして付ける必要があります。例えば、すべての Amazon SNS アクションを指定するには、`sns:*` を使用します。アクションのリストについては、「Amazon Simple Notification Service API リファレンス」にアクセスしてください。

Amazon SNS ポリシーキー

Amazon SNS は、以下の AWS 全体のポリシーキーに加えて、いくつかのサービス固有のキーを実装しています。

各 AWS のサービス サービスによってサポートされている条件キーのリストについては、IAM ユーザーガイドの AWS のサービス サービスのアクション、リソース、条件キーを参照してください。複数の AWS のサービス サービスで使用できる条件キーのリストについては、IAM ユーザーガイドのAWS グローバル条件コンテキストキーを参照してください。

Amazon SNS では以下のサービス固有のキーを使用します。`Subscribe` リクエストへのアクセスを制限するポリシーで、これらのキーを使用します。

- sns:Endpoint - Subscribe リクエストからの URL、E メールアドレス、または ARN、あるいは以前に確認されたサブスクリプション。文字列条件とともに使用 (「Amazon SNS のポリシーの例」を参照) して特定のエンドポイント (*@yourcompany.com など) へのアクセスを制限します。
- sns:Protocol - Subscribe リクエストまたは以前に確認されたサブスクリプションからの protocol 値。文字列条件とともに使用 (「Amazon SNS のポリシーの例」を参照) して、特定の配信プロトコル (https など) への発行を制限します。

Amazon SNS のポリシーの例

このセクションでは、Amazon SNS へのユーザーアクセスをコントロールするための簡単なポリシーをいくつか紹介します。

> **Note**
>
> 将来的には、Amazon SNS に新しいアクションが追加される可能性があります。これらは、以下のポリシーのいずれかに論理的に含まれ、ポリシーに記載された目的に基づいている必要があります。

Example 1: グループでトピックの作成と管理を許可する

この例では、CreateTopic、ListTopics、SetTopicAttributes、および DeleteTopic へのアクセスを付与するポリシーを作成します。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2: IT グループが特定のトピックにメッセージを発行することを許可する

この例では、IT のグループを作成し、対象の特定のトピックで Publish へのアクセスを付与するポリシーを割り当てます。

```
{
  "Statement": [{
```

```
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3: AWS アカウント のユーザーが、トピックを受信登録できるようにする

この例では、sns:Protocol および sns:Endpoint ポリシーキーの文字列一致条件を使って Subscribe アクションにアクセスを付与するポリシーを作成します。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "SNS:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4: パートナーが特定のトピックにメッセージを発行することを許可する

Amazon SNS ポリシーまたは IAM ポリシーを使用して、パートナーが特定のトピックに発行することを許可できます。パートナーに AWS アカウント がある場合は、Amazon SNS ポリシーを使用した方が簡単になる可能性があります。ただし、AWS セキュリティ認証情報を所有するパートナー会社のユーザーは、トピックにメッセージを発行できます。この例では、特定のユーザー (またはアプリケーション) へのアクセスを制限するとします。そのためには、パートナーを自社内のユーザーのように扱い、Amazon SNS ポリシーの代わりに IAM ポリシーを使用する必要があります。

この例では、パートナー会社 WidgetCo を表す というグループを作成し、アクセスが必要なパートナー会社で特定の人 (またはアプリケーション) のユーザーを作成し、そのユーザーをグループに入れます。

次に、 という名前の特定のトピックに対するPublishアクセス許可をグループに付与するポリシーをアタッチしますWidgetPartnerTopic。

また、 WidgetCo グループがトピックに対して他の操作を実行できないようにしたいため、 以外のトピックPublishに対する 以外の Amazon SNS アクションへのアクセス許可を拒否するステートメントを追加します WidgetPartnerTopic。これは、システム内の他の場所に、Amazon SNS への幅広いアクセスをユーザーに付与する広範なポリシーが存在する場合にのみ必要です。

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

Amazon SNS で一時的なセキュリティ認証情報を使用する

IAM では、IAM ユーザーが独自のセキュリティ認証情報を使用して作成されるのに加えて、AWS サービスおよびリソースへのアクセスを許可するユーザーに一時的なセキュリティ認証情報を付与することもできます。AWS アカウント を持つユーザー (つまり、IAM ユーザー) を管理できます。AWS アカウント を持たないシステムのユーザー (つまり、統合ユーザー) を管理することもできます。さらに「ユーザー」を、AWS リソースにアクセスするために作成するアプリケーションにすることもできます。

Amazon SNS にリクエストするときに、これらの一時的なセキュリティ認証情報を使用できます。APIライブラリによって、これらの認証情報を使用して必要な署名値が計算されて、リクエストが認証されます。失効した証明書を使用してリクエストを送信した場合、Amazon SNS はリクエストを拒否します。

IAM による一時的なセキュリティ認証情報のサポートについては、『IAM の使用』の「AWS リソースへの一時的なアクセス権の付与」を参照してください。

Example 一時的なセキュリティ認証情報を使用した Amazon SNS リクエストの認証

以下の例は、一時的なセキュリティ認証情報を取得して Amazon SNS リクエストを認証する方法を示しています。

```
http://sns.us-east-2.amazonaws.com/
?Name=My-Topic
&Action=CreateTopic
&Signature=gfzIF53exFVdpSNb8AiwN3Lv%2FNYXh6S%2Br3yySK70oX4%3D
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-03-31T12%3A00%3A00.000Z
&SecurityToken=SecurityTokenValue
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

Amazon SNS API のアクセス許可: アクションとリソースのリファレンス

次のリストは、Amazon SNS でのアクセスコントロールの実装に固有の情報を示しています。

- 各ポリシーは、1 つのトピックだけを対象とする必要があります (ポリシーを作成する際は、異なるトピックを対象とするステートメントを含めないでください)。
- 各ポリシーには固有のポリシー (Id) が必要です。
- ポリシーを構成する各ステートメントには固有のステートメント (sid) が必要です。

ポリシーのクォータ

以下の表では、ポリシーステートメントの最大クォータを示しています。

名前	最大クォータ
バイト	30 KB
ステートメント	100
プリンシパル	1 ~ 200 (0 は無効)
リソース	1 (0 は無効です。値はポリシーのトピックの ARN と一致する必要があります)

有効な Amazon SNS ポリシーアクション

Amazon SNS は、次の表に示すアクションをサポートします。

[アクション]	説明
sns:AddPermission	トピックポリシーへのアクセス権限の追加を許可します。
sns:DeleteTopic	トピックを削除する許可を与えます。
sns:GetDataProtectionPolicy	トピックのデータ保護ポリシーを返すアクセス許可を付与
sns:GetTopicAttributes	すべてのトピック属性を受け取る許可を与えます。
sns:ListSubscriptionsByTopic	特定のトピックへのサブスクリプションをすべて取得することを許可します。
sns:ListTagsForResource	指定したトピックに追加されたすべてのタグを一覧表示する許可を与えます。
sns:Publish	トピックやエンドポイントへの公開とバッチの公開の両方に対する許可を付与します。詳細については、「Amazon Simple Notification Service API リファレンスPublishBatch」の「 および を公開する」を参照してください。
sns:PutDataProtectionPolicy	トピックのデータ保護ポリシーを設定するアクセス許可を付与
sns:RemovePermission	トピックポリシーのアクセス権限を削除する許可を与えます。
sns:SetTopicAttributes	トピックの属性を設定する許可を与えます。
sns:Subscribe	トピックにサブスクライブすることを許可します。

サービス固有のキー

Amazon SNS では以下のサービス固有のキーを使用します。これらは、`Subscribe` リクエストへのアクセスを制限するポリシーに使用できます。

- sns:endpoint - `Subscribe` リクエストからの URL、E メールアドレス、または ARN、あるいは以前に確認されたサブスクリプション。文字列条件とともに使用 (「Amazon SNS のポリシーの例」を参照) して、特定のエンドポイント (*@example.com など) へのアクセスを制限します。

- sns:Protocol - `Subscribe` リクエストまたは以前に確認されたサブスクリプションからの `protocol` 値。文字列条件とともに使用 (「Amazon SNS のポリシーの例」を参照) して、特定の配信プロトコル (https など) への発行を制限します。

> **Important**
>
> sns:Endpoint によるアクセスの制御にポリシーを使用するときは、将来、DNS の問題がエンドポイントの名前解決に影響する可能性があるため、注意が必用です。

Amazon Simple Notification Service アイデンティティとアクセスのトラブルシューティング

Amazon SNS と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復には、次の情報を利用してください。

トピック

- Amazon SNS でアクションを実行する認可がありません
- iam を実行する権限がありません。PassRole
- 自分の AWS アカウント 以外のユーザーに Amazon SNS リソースへのアクセスを許可したい

Amazon SNS でアクションを実行する認可がありません

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

以下のエラー例は、`mateojackson` ユーザーがコンソールを使用して架空の *`my-example-widget`* リソースに関する詳細情報を表示しようとしているが、架空の `sns:`*`GetWidget`* 許可がないという場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

この場合、Mateo のポリシーでは、*`my-example-widget`* アクションを使用して `sns:`*`GetWidget`* リソースへのアクセスを許可するように更新する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。管理者とは、サインイン認証情報を提供した担当者です。

iam を実行する権限がありません。PassRole

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon SNS にロールを渡せるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールやサービスリンクロールを作成せずに、既存のロールをサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次の例のエラーは、`marymajor` という IAM ユーザーがコンソールを使用して Amazon SNS でアクションを実行しようとする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新して Mary に `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。管理者とは、サインイン認証情報を提供した担当者です。

自分の AWS アカウント 以外のユーザーに Amazon SNS リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセス制御リスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- Amazon SNS がこれらの機能をサポートしているかどうかを確認するには、「Amazon Simple Notification Service で IAM を使用する方法」を参照してください。
- 所有している AWS アカウント 全体のリソースへのアクセス権を提供する方法については、『IAM ユーザーガイド』の「所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに提供」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「サードパーティーが所有する AWS アカウント にアクセス権を提供する」を参照してください。

- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

Amazon SNS でのログ記録とモニタリング

このセクションでは、Amazon SNS トピックのログ作成およびモニタリングについて説明します。

トピック

- CloudTrail を使用した Amazon SNS API コールのログ記録
- CloudWatch を使用した Amazon SNS のモニタリング

CloudTrail を使用した Amazon SNS API コールのログ記録

Amazon SNS は AWS CloudTrail と統合されています。このサービスは、ユーザーやロール、または Amazon SNS の AWS のサービスによって実行されたアクションを記録するサービスです。CloudTrail は、Amazon SNS の API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、Amazon SNS コンソールからの呼び出しと、Amazon SNS API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、Amazon S3 のイベントなど、Amazon SNS バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [Event history] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、Amazon SNS に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

設定や有効化の方法など、CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail の Amazon SNS 情報

CloudTrail は、アカウント作成時に AWS アカウント で有効になります。Amazon SNS でサポートされているイベントアクティビティが発生すると、そのアクティビティは [Event history] の他の AWS のサービスのイベントとともに CloudTrail イベントに記録されます。最近のイベントは、AWS アカウント で表示、検索、ダウンロードできます。詳細については、「CloudTrail イベント履歴でイベントを表示する」を参照してください。

Amazon SNS のイベントなどの、AWS アカウント におけるイベントを継続的に記録するには、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべての AWS リージョンに適用されます。追跡は、AWSパーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- 追跡を作成するための概要
- CloudTrail のサポート対象サービスと統合
- Amazon SNS の CloudTrail の通知の設定
- 「複数のリージョンから CloudTrail ログファイルを受け取る」および「複数のアカウントから CloudTrail ログファイルを受け取る」

CloudTrail のコントロールプレーンイベント

Amazon SNS は、CloudTrail ログファイルのイベントとして以下のアクションのログ付けをサポートします。

- 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**
>
> Amazon Web Services にログインしていないとき (非認証モード)、ConfirmSubscription または Unsubscribe アクションが呼び出されても、そのアクションは CloudTrail に記録されま

せん。例えば、E メール通知内のリンクを選択して、あるトピックへの保留中のサブスクリプションを確認したとき、ConfirmSubscription アクションが非認証モードで呼び出されたとします。この場合、ConfirmSubscription アクションは CloudTrail に記録されません。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストが、ルート認証情報と AWS Identity and Access Management (IAM) ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーティッドユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

CloudTrail のデータプレーンイベント

CloudTrail ファイルで以下の API アクションのロギングを有効にするには、CloudTrail でデータプレーン API アクティビティのログ記録を有効にする必要があります。詳細については、「AWS CloudTrail ユーザーガイド」の「データイベントをログ記録する」を参照してください。

データプレーンイベントは、リソースタイプ別にフィルタリングすることもできます。これにより、どの DynamoDB API コールを選択して CloudTrail でログに記録し、支払うかをきめ細かく制御できます。例えば、AWS::SNS::Topic をリソースタイプとして指定することで、トピックの Publish や PublishBatch API アクションへのコールをログに記録できます。同様に、AWS::SNS::PlatformEndpoint をリソースタイプとして指定することで、プラットフォームエンドポイントの Publish API アクションへのコールをログに記録できます。詳細については、「AWS CloudTrail API リファレンス」の「AdvancedEventSelector」を参照してください。

> **Note**
>
> Amazon SNS リソースタイプ AWS::SNS::PhoneNumber は、CloudTrail によってログに記録されません。

Amazon SNS データプレーン API

- Publish
- PublishBatch

例: Amazon SNS ログファイルのエントリ

[トレイル] は、指定した Simple Storage Service (Amazon S3) バケットにイベントをログファイルとして配信するように構成できます。CloudTrail のログファイルには、単一か複数のログエントリがあります。イベントはあらゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次は、ListTopics、CreateTopic、および DeleteTopic のアクションを示す CloudTrail ログエントリの例です。


```
{
  "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"
    },
  ]
}
```

以下の例は、Publish アクションと PublishBatch アクションを示す CloudTrail ログエントリです。

発行

```
{
 "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"
 }
}
```

CloudWatch を使用した Amazon SNS のモニタリング

Amazon SNS と Amazon CloudWatch は統合されているため、個々のアクティブな Amazon SNS 通知についてメトリクスを収集、表示、分析できます。Amazon SNS に対して CloudWatch を設定すると、 トピック、プッシュ通知、および SMS 配信のパフォーマンスをより正確に把握できます。例えば、NumberOfNotificationsFailed など、Amazon SNS メトリクスの指定のしきい値に達した場合に E メール通知が送信されるよう、アラームを設定することができます。Amazon SNS が CloudWatch に送信するすべてのメトリクスのリストについては、「Amazon SNS のメトリクス」を参照してください。Amazon SNS プッシュ通知の詳細については、「モバイルプッシュ通知」を参照してください。

> **Note**
>
> CloudWatch を使用して Amazon SNS トピック用に設定するメトリクスは、自動的に収集され、1 分間隔で CloudWatch にプッシュ通知されます。これらのメトリックスは、CloudWatch ガイドラインを満たすすべてのトピックで収集され、アクティブになります。CloudWatch は、トピックの最後のアクティビティ (つまり、任意の API コール) から最大 6 時間、そのトピックをアクティブと見なします。

CloudWatch でレポートされた Amazon SNS メトリクスに対して料金は発生しません。それらは Amazon SNS サービスの一部として提供されます。

Amazon SNS 用の CloudWatch メトリクス

Amazon SNS のメトリクスは CloudWatch の独自のコマンドインターフェイス (CLI) を使用して、あるいはプログラムによって CloudWatch API を使用してモニタリングできます。次の手順は、AWS Management Console を使用してメトリクスにアクセスする方法を示しています。

CloudWatch コンソールを使用してメトリクスを表示するには

1. CloudWatch コンソールにサインインします。
2. ナビゲーションパネルで [Metrics] を選択します。
3. [All metrics] タブで [SNS] を選択し、次のいずれかのディメンションを選択します。
   - 国/地域、SMS タイプ
   - 電話番号
   - トピックのメトリクス
   - ディメンションの定められていないメトリクス
4. 詳細を表示するには、特定の項目を選択します。例えば、[Topic Metrics] を選択し、[NumberOfMessagesPublished] を選択すると、6 時間の時間範囲における 1 分間の公開済み Amazon SNS メッセージの平均数が表示されます。
5. Amazon SNS の使用状況メトリクスを表示するには、[All metrics] (すべてのメトリクス) タブで [Usage] (使用率) を選択し、対象の Amazon SNS 使用率メトリクス (例: `NumberOfMessagesPublishedPerAccount`) を選択します。

Amazon SNS メトリクス用の CloudWatch メトリクスを設定する

CloudWatch では、メトリックスのしきい値に到達したときのアラームを設定することもできます。例えば、サンプリング期間内に指定されたしきい値に到達した場合、イベントについて知らせる E メール通知が送信されるように、メトリクス NumberOfNotificationsFailed のアラームを設定できます。

CloudWatch コンソールを使用してアラームを設定するには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. [Alarms]、[Create Alarm] の順に選択します。これにより、[Create Alarm] ウィザードが起動します。
3. Amazon SNS メトリクスをスクロールして、アラームを設定するメトリクスを見つけます。アラームを設定するメトリクスを選択してから、[Continue] を選択します。
4. [Name]、[Description]、[Threshold]、[Time] のそれぞれにメトリクスの値を入力し、[Continue] を選択します。
5. アラーム状態として [Alarm] を選択します。アラーム状態になったときに CloudWatch から E メールが届くようにするには、既存の Amazon SNS トピックを選択するか、[新しいメールトピックの作成] を選択します。[新しいメールトピックの作成] を選択した場合は、新しいトピックの名前と E メールアドレスを設定できます。このリストは、今後のアラーム用に保存され、ドロップダウンボックスに表示されます。[Continue] (続行) をクリックします。

   > **Note**
   >
   > [Create New Email Topic] を使用して新しい Amazon SNS トピックを作成する場合、メールアドレスを検証しなければ、そのアドレスで通知を受け取ることができません。E メールは、アラームがアラーム状態になったときにのみ送信されます。アラーム状態になったときに、メールアドレスの検証がまだ完了していない場合は、そのアドレスで通知を受け取ることはできません。

6. この時点で、[Create Alarm] ウィザードで、作成するアラームを確認できます。何らかの変更を行う必要がある場合は、右側にある [Edit] リンクを使用します。希望どおりの設定になったら、[Create Alarm] を選択します。

CloudWatch とアラームの使用方法の詳細については、「CloudWatch のドキュメント」を参照してください。

Amazon SNS のメトリクス

Amazon SNS は、次のメトリクスを CloudWatch に送信します。

名前空間	メトリクス	説明
AWS/SNS	NumberOfMessagesPublished	Amazon SNS トピックに対して発行されたメッセージ数。 単位: Count 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum
AWS/SNS	NumberOfNotificationsDelivered	Amazon SNS トピックからそのトピックにサブスクライブしているエンドポイントに正常に配信されたメッセージ数。 配信の試行が成功するには、エンドポイントのサブスクリプションでメッセージが許可されている必要があります。サブスクリプションでメッセージが許可されるのは、a.) フィルターポリシーがないか、b.) フィルターポリシーに、メッセージに割り当てられた属性と一致する属性が含まれる場合です。サブスクリプションでメッセージが拒否された場合、配信の試行はこのメトリクスに対してカウントされません。 単位: Count 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum

名前空間	メトリクス	説明
AWS/SNS	NumberOfNotificationsFailed	Amazon SNS が配信に失敗したメッセージの数。 Amazon SQS、E メール、SMS、またはモバイルプッシュエンドポイントの場合、Amazon SNS がメッセージ配信の試行を停止すると、このメトリクスが 1 増分されます。HTTP または HTTPS エンドポイントの場合、失敗した配信の試行がすべて、初回試行に続く再試行を含め、このメトリクスに対してカウントされます。その他のすべてのエンドポイントの場合、メッセージが配信されないと、カウントが 1 増加します (試行回数には関係ありません)。 サブスクリプションフィルターポリシーによって拒否されたメッセージは、このメトリクスに対してカウントされません。 HTTP エンドポイントの再試行回数をコントロールできます。詳細については、「Amazon SNS メッセージ配信の再試行」を参照してください。 単位: Count 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum、Average

名前空間	メトリクス	説明
AWS/SNS	NumberOfNotificationsFilteredOut	サブスクリプションフィルターポリシーによって拒否されたメッセージの数。メッセージの属性がポリシーの属性と一致しない場合、フィルターポリシーによってメッセージは拒否されます。 単位: Count 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum、Average
AWS/SNS	NumberOfNotificationsFilteredOut-MessageAttributes	属性ベースのフィルタリングのサブスクリプションフィルターポリシーによって拒否されたメッセージの数。 単位: CountValid ディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum、Average

名前空間	メトリクス	説明
AWS/SNS	NumberOfNotificationsFilteredOut-MessageBody	ペイロードベースのフィルタリングのサブスクリプションフィルターポリシーによって拒否されたメッセージの数。 単位はカウント 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum、Average
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidAttributes	メッセージの属性が無効であるため (例えば属性の JSON 形式が正しくないため)、サブスクリプションフィルターポリシーによって拒否されたメッセージの数。 単位: Count 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum、Average

名前空間	メトリクス	説明
AWS/SNS	NumberOfNotificationsFilteredOut-NoMessageAttributes	メッセージに属性がないため、サブスクリプションフィルターポリシーによって拒否されたメッセージの数。 単位: Count 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum、Average
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidMessageBody	メッセージ本文がフィルタリングに対して無効であるため (JSON メッセージ本文が無効であるなど)、サブスクリプションフィルターポリシーによって拒否されたメッセージの数。 単位はカウント 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum、Average
AWS/SNS	NumberOfNotificationsRedrivenToDlq	デッドレターキューに移動されたメッセージの数。 単位: Count 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum、Average

名前空間	メトリクス	説明
AWS/SNS	NumberOfNotificationsFailedToRedriveToDlq	デッドレターキューに移動できなかったメッセージの数。 単位: Count 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Sum、Average
AWS/SNS	PublishSize	発行されたメッセージのサイズ。 単位: Bytes 有効なディメンション: アプリケーション、PhoneNumber、プラットフォーム、TopicName 有効な統計: Minimum、Maximum、Average、Count

名前空間	メトリクス	説明
AWS/SNS	SMSMonthToDateSpentUSD	今月の始めから今日までの SMS メッセージの送信料金。 今月の始めから今日までの料金がアカウントの毎月の SMS 使用限度に近付いたことがわかるように、このメトリクスにアラームを設定できます。Amazon SNS が、SMS メッセージを送信するとこの限度を超えるコストが発生すると判断する場合、数分以内に SMS メッセージの発行を停止します。 SMS の毎月の使用料限度の設定の詳細、または AWS に対して限度の引き上げをリクエストする方法については、「SMS メッセージプリファレンスを設定する」を参照してください。 単位: USD 有効なディメンション: PhoneNumber 有効な統計: Maximum
AWS/SNS	SMSSuccessRate	正常な SMS メッセージ配信のレート 単位: Count 有効なディメンション: PhoneNumber 有効な統計: Sum、Average、Data Samples

Amazon SNS メトリクスのディメンション

Amazon Simple Notification Service は、以下のディメンションを CloudWatch に送信します。

ディメンション	説明
Application	アプリケーションオブジェクトのフィルター。APN や FCM など、サポートされるプッシュ通知サービスの 1 つに登録されているアプリケーションやデバイスを表します。
Application,Platform	アプリケーションおよびプラットフォームオブジェクトのフィルター。プラットフォームオブジェクトは、APN や FCM など、サポートされるプッシュ通知サービスで使用されます。
Country	SMS メッセージの送信先の国またはリージョンのフィルター。国またはリージョンは、ISO 3166-1 alpha-2 コードで表されます。
PhoneNumber	SMS を電話番号に直接公開するときに、電話番号にフィルターをかけます (トピックなし)。
Platform	APN や FCM などのプッシュ通知サービスで使用されるプラットフォームオブジェクトのフィルター。
TopicName	Amazon SNS トピック名のフィルター。
SMSType	SMS メッセージのメッセージタイプのフィルター。プロモーションまたはトランザクションがあります。

Amazon SNS 使用率メトリクス

Amazon Simple Notification Service は、以下の使用状況メトリクスを CloudWatch に送信します。

名前空間	サービス	メトリクス	リソース	タイプ	説明
AWS/使用	SNS	ResourceCount	NumberOfMessagesPu	リソース	• AWS アカウント全体で Amazon

名前空間	サービス	メトリクス	リソース	タイプ	説明
			blishedPerAccount		SNS トピックに対して発行されたメッセージ数。 • 単位: なし • 有効な統計: Sum
AWS/使用	SNS	ResourceCount	ApproximateNumberOfTopics	リソース	• AWS アカウント全体のトピックのおおよその数。 • 単位: なし • 有効な統計: Average、Minimum、Maximum、Sum
AWS/使用	SNS	ResourceCount	ApproximateNumberOfFilterPolicies	リソース	• AWS アカウント全体のフィルターポリシーのおおよその数。 • 単位: なし • 有効な統計: Average、Minimum、Maximum、Sum

名前空間	サービス	メトリクス	リソース	タイプ	説明
AWS/使用	SNS	ResourceCount	ApproximateNumberOfPendingSubscriptions	リソース	• AWS アカウント全体の保留中のサブスクリプションのおおよその数。 • 単位: なし • 有効な統計: Average、Minimum、Maximum、Sum

名前空間	サービス	メトリクス	リソース	タイプ	説明
AWS/使用	SNS	CallCount	• AddPermission • CheckIfPhoneNumberIsOptedOut • CreatePlatformApplication • CreatePlatformEndpoint • ConfirmSubscription • CreateSMSSandboxPhoneNumber • CreateTopic • DeleteEndpoint • DeletePlatformApplication • DeleteSMSSandboxPhoneNumber • DeleteTopic	API	• AWS アカウント全体で選択した Amazon SNS API の API コールの数。 • 単位: なし • 有効な統計: Sum

名前空間	サービス	メトリクス	リソース	タイプ	説明
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

名前空間	サービス	メトリクス	リソース	タイプ	説明
			• ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes • SetSubscriptionAttributes • SetTopicAttributes		

名前空間	サービス	メトリクス	リソース	タイプ	説明
			• Subscribe • Unsubscribe • UntagResource • VerifySMSSandboxPhoneNumber		

Amazon SNS のコンプライアンス検証

サードパーティーの監査者は、Health Insurance Portability and Accountability Act (HIPAA) を含む複数の AWS コンプライアンスプログラムの一環として、Amazon SNS のセキュリティおよびコンプライアンスを評価します。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスプログラムによる対象範囲内の AWS サービス」を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」「」「」を参照してください。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。詳細については、AWS Artifactにおけるレポートのダウンロード を参照してください。

Amazon SNS を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性や貴社のコンプライアンス目的、適用される法律および規制によって決まります。AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティ & コンプライアンスクイックスタートガイド – これらのデプロイガイドには、アーキテクチャ上の考慮事項の説明と、AWS でセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするためのステップが記載されています。
- HIPAA セキュリティおよびコンプライアンスホワイトペーパーのアーキテクチャの設計 - このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。

- AWS コンプライアンスのリソース - このワークブックおよびガイドのコレクションは、ユーザーの業界や地域で使用できるかもしれません。
- AWS Config デベロッパーガイドのルールでのリソースの評価 - AWS Config サービスでは、自社のプラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態を評価します。
- AWS Security Hub - この AWS サービスは、AWS 内でのユーザーのセキュリティ状態に関する包括的な見解を提供し、業界のセキュリティスタンダード、およびベストプラクティスに対するコンプライアンスを確認するために役立ちます。

Amazon SNS の耐障害性

Amazon SNS の耐障害性は、 AWS リージョン およびアベイラビリティーゾーンを中心に展開される AWS グローバルインフラストラクチャを活用することで保証されます。また、低レイテンシー、高スループット、および高度冗長ネットワークで接続された物理的に独立および隔離されたアベイラビリティーゾーン AWS リージョン を提供します。このアーキテクチャにより、中断することなくアベイラビリティーゾーン間でシームレスにフェイルオーバーできるため、アプリケーションとデータベースは従来のデータセンターインフラストラクチャに比べて本質的に耐障害性とスケーラビリティが向上します。アベイラビリティーゾーンを使用することで、Amazon SNS サブスクライバーは可用性と信頼性の向上から利点を得られるため、中断の可能性があるメッセージ配信が保証されます。 AWS リージョン およびアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

さらに、Amazon SNS トピックへのサブスクリプションは、配信の再試行とデッドレターキューで設定できるため、一時的な障害の自動処理が可能になり、メッセージが意図した宛先に確実に到達できるようになります。

Amazon SNS は、メッセージフィルタリングとメッセージ属性もサポートしています。これにより、特定のユースケースに合わせて回復力戦略を調整できるため、アプリケーションの全体的な堅牢性が向上します。

Amazon SNS のインフラストラクチャセキュリティ

マネージドサービスである Amazon SNS は、「セキュリティ、アイデンティティ、コンプライアンスのベストプラクティス」ドキュメントに記載されている AWS グローバルネットワークセキュリティ手順で保護されています。

AWS API アクションを使用して、ネットワーク経由で Amazon SNS にアクセスします。クライアントは、Transport Layer Security (TLS) 1.2 以降をサポートする必要があります。また、Ephemeral Diffie-Hellman (DHE)やElliptic Curve Ephemeral Diffie-Hellman(ECDHE)などの Perfect Forward Secrecy(PFS)を使用した暗号スイートもクライアントでサポートされている必要があります。

IAM プリンシパルに関連付けられているアクセスキー ID とシークレットアクセスキーの両方を使用してリクエストに署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

これらの API アクションは任意のネットワークの場所から呼び出すことができますが、Amazon SNS ではリソースベースのアクセスポリシーがサポートされています。これには送信元 IP アドレスに基づく制限を含めることができます。また、Amazon SNS ポリシーを使用して、特定の Amazon VPC エンドポイントまたは特定の VPC からのアクセスを制御することもできます。これにより、ネットワーク内の特定の VPC からのみ特定の Amazon SNS トピックへの AWS ネットワークアクセスが効果的に分離されます。詳細については、「Amazon SNS トピックの発行を特定の VPC エンドポイントのみからに制限する」を参照してください。

Amazon SNS のセキュリティベストプラクティス

AWS には、Amazon SNS の多くのセキュリティ機能があります。独自のセキュリティポリシーのコンテキストで、これらのセキュリティ機能を確認します。

> **Note**
>
> これらのセキュリティ機能のガイダンスは、一般的ユースケースと実装に適用されます。特定のユースケース、アーキテクチャ、脅威モデルのコンテキストで、これらのベストプラクティスを確認することをお勧めします。

予防的ベストプラクティス

以下に、Amazon SNS の予防的なセキュリティに関するベストプラクティスを示します。

トピック

- トピックがパブリックアクセス可能でないようにする
- 最小特権アクセスの実装
- Amazon SNS アクセスを必要とするアプリケーションと AWS のサービスには IAM ロールを使用する。

- サーバー側の暗号化を実装する
- 送信時のデータの暗号化を強制する
- VPC エンドポイントを使用して Amazon SNS にアクセスすることを検討する
- サブスクリプションが raw http エンドポイントに配信するように設定されていないことを確認する

トピックがパブリックアクセス可能でないようにする

インターネット上の誰もが Amazon SNS トピックに読み書きできるように明示的にリクエストしない限り、トピックにパブリックアクセスできないようにする必要があります (世界中の誰でも、または認証された AWS ユーザーがアクセス可能)。

- `Principal`を`""`に設定してポリシーを作成しないでください。
- ワイルドカード (*) を使用しないでください。代わりに、特定のユーザーに名前を付けます。

最小特権アクセスの実装

アクセス権限を付与する場合、アクセス権限を受け取るユーザー、アクセス許可の対象となるトピック、およびこれらのトピックに対して許可する特定の API アクションを決定します。最小権限の原則を実装することは、セキュリティリスクを軽減するために重要です。また、エラーや悪意のある意図による悪影響を減らすのにも役立ちます。

最小特権を付与するスタンダードのセキュリティアドバイスに従ってください。つまり、特定のタスクの実行に必要なアクセス権限のみを付与します。ユーザーアクセスに関連するセキュリティポリシーを組み合わせて使用することで、最小権限を実装できます。

Amazon SNS では、発行者と受信者のモデルが使用され、次の 3 種類のユーザーアカウントアクセスが必要です。

- 管理者 - トピックの作成、変更、削除にアクセスします。管理者は、トピックポリシーも制御します。
- 発行者 - トピックへのメッセージ送信のアクセス権限を持ちます。
- 受信者 - トピックへの登録のアクセス権限を持ちます。

詳細については、次のセクションを参照してください。

- Amazon SNS での Identity and Access Management

- Amazon SNS API のアクセス許可: アクションとリソースのリファレンス

Amazon SNS アクセスを必要とするアプリケーションと AWS のサービスには IAM ロールを使用する。

Amazon SNS トピックにアクセスするアプリケーションまたは Amazon EC2 などの AWS のサービスには、AWS API リクエストで有効な AWS 認証情報を使用する必要があります。これらの認証情報は自動的には更新されないため、AWS 認証情報をアプリケーションまたは EC2 インスタンスに直接保存しないでください。

代わりに、IAM ロールを使用して、Amazon SNS にアクセスする必要があるアプリケーションまたはサービスの一時的な認証情報を管理することをおすすめします。ロールを使用するとき、EC2 インスタンスまたは AWS のサービス (AWS Lambda など) に長期の認証情報 (ユーザー名、パスワード、アクセスキーなど) を配布する必要はありません。代わりに、ロールは、アプリケーションが他の AWS リソースへの呼び出しを行うときに使用できる一時的なアクセス権限を提供します。

詳細については、「IAM ユーザーガイド」の「IAM ロール」および「ロールの一般的なシナリオ: ユーザー、アプリケーション、およびサービス」を参照してください。

サーバー側の暗号化を実装する

データ漏洩の問題を軽減するには、保存時の暗号化を使用して、メッセージを保存する場所とは別の場所に保存されているキーを使用してメッセージを暗号化します。サーバー側の暗号化 (SSE) は、保存時のデータ暗号化を提供します。Amazon SNS は、データを保存するときにメッセージレベルで暗号化し、アクセスするとメッセージを復号します。SSEはAWS Key Management Serviceでマネージドされているキーを使用します。リクエストが認証され、お客様がアクセス権限を持っていれば、トピックが暗号化されているかどうかに関係なく同じ方法でアクセスできます。

詳細については、保管中の暗号化 および キーの管理 を参照してください。

送信時のデータの暗号化を強制する

HTTP を使用して送信中に暗号化されていないメッセージを公開することは可能ですが、お勧めできません。ただし、暗号化された SNS トピックに公開するときに HTTP を使用することはできません。

AWS では、HTTP ではなく HTTPS を使用することをお勧めします。HTTPS を使用すると、SNS トピック自体が暗号化されていなくても、メッセージは送信中に自動的に暗号化されます。HTTPS

を使用しない場合、ネットワークベースの攻撃者は、中間者などの攻撃を使用して、ネットワークトラフィックを傍受したり操作することができます。

HTTPS 経由の暗号化された接続のみを実行するには、aws:SecureTransport 条件を、暗号化されていない SNS トピックに添付されている IAM ポリシーに追加します。これにより、メッセージ発行者は HTTP ではなく HTTPS を使用することになります。次の例のポリシーをガイドとして使用できます。

```
{
  "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": "*"
    }
  ]
}
```

VPC エンドポイントを使用して Amazon SNS にアクセスすることを検討する

操作できる必要があるが、インターネットに絶対に公開してはならないトピックがある場合は、VPC エンドポイントを使用して、特定の VPC 内のホストのみにトピックアクセスを制限します。トピックポリシーを使用して、特定の VPC エンドポイントまたは特定の VPC からのトピックへのアクセスを制御できます。

Amazon SNS の VPC エンドポイントには、メッセージへのアクセスを制御するために、2 通りの方法が用意されています。

- 特定の VPC エンドポイントを通じて許可されるリクエスト、ユーザー、またはグループを管理できます。

- トピックポリシーを使用して、どの VPC または VPC エンドポイントがトピックにアクセスできるかを制御できます。

詳細については、エンドポイントの作成 および Amazon SNS 用の VPC エンドポイントポリシーを作成する を参照してください。

サブスクリプションが raw http エンドポイントに配信するように設定されていないことを確認する

サブスクリプションは raw http エンドポイントに配信する設定をしないでください。エンドポイントドメイン名に配信するサブスクリプションが必ずなければいけません。例えば、エンドポイントに配信するように構成されたサブスクリプションである `http://1.2.3.4/my-path` は `http://my.domain.name/my-path` に変更する必要があります。

Amazon SNS トピックのトラブルシューティング

このセクションでは、Amazon SNS トピックのトラブルシューティングについて説明します。

AWS X-Ray を使用した Amazon SNS トピックのトラブルシューティング

AWS X-Ray は、アプリケーションが処理するリクエストに関するデータを収集し、データの表示とフィルター処理を可能にして、潜在的な問題や最適化の機会を識別できるようにします。アプリケーションに対するトレース対象のリクエストの場合、リクエスト、レスポンス、およびアプリケーションがダウンストリーム AWSリソース、マイクロサービス、データベース、HTTPウェブAPI に対して行う呼び出しの詳細な情報を表示できます。

Amazon SNS で X-Ray を使用して、アプリケーションを通過するメッセージをトレースおよび分析できます。AWS Management Console を使用して、Amazon SNS およびアプリケーションが使用する他のサービス間の接続マップを表示できます。コンソールを使用して、平均レイテンシーや障害発生率などのメトリクスを表示することもできます。詳細については、『AWS X-Ray デベロッパーガイド』の「Amazon SNS と AWS X-Ray」を参照してください。

Amazon SNS でのアクティブトレース

を使用してAWS X-Ray、ユーザーリクエストが Amazon SNS トピックを経由して Amazon Data Firehose 、、Amazon SQSAWS Lambda、および HTTP/S エンドポイントサブスクリプションに移動するときに、ユーザーリクエストを追跡および分析できます。X-Ray はリクエスト全体 end-to-end を表示するため、Amazon SNS トピックを呼び出しているものと、トピックのサブスクリプションの下流にあるものを表示できます。メッセージとそのバックエンドサービスのレイテンシーを分析できます (例えば、リクエストがトピックに費やされる時間や、トピックの各サブスクリプションにメッセージを配信するのにかかった時間など)。

> **Important**
>
> 多数のサブスクリプションがある Amazon SNS トピックはサイズ制限に達し、完全にトレースされない場合があります。トレースドキュメントのサイズ制限については、AWS 全般のリファレンスの「X-Ray サービスクォータ」を参照してください。

すでにトレースされているサービスから Amazon SNS API を呼び出すと、Amazon SNS は、API で X-Ray トレースが有効になっていなくてもトレースをパススルーします。

Amazon SNS は、標準トピックと FIFO トピックの両方で X-Ray トレースをサポートしています。Amazon SNS トピックの X-Ray を有効にするには、Amazon SNS コンソール、Amazon SNS `SetTopicAttributes` API、Amazon Simple 通知サービス CLI リファレンス、または AWS CloudFormation を使用します。

Amazon SNS で X-Ray を使用する方法の詳細については、「AWS X-Ray デベロッパーガイド」の「Amazon SNS と AWS X-Ray」を参照してください。

トピック

アクティブトレースのアクセス許可

Amazon SNS コンソールを使用する場合、Amazon SNS は Amazon SNS トピックが X-Ray を呼び出すために必要なアクセス許可を作成しようとします。Amazon SNS コンソールを使用するための十分なアクセス許可がない場合、この試行は拒否される可能性があります。詳細については、「Amazon SNS での Identity and Access Management」および「Amazon SNS アクセスコントロールのケース例」を参照してください。

CLI を使用するときは、アクセス許可を手動で設定する必要があります。これらのアクセス許可は、リソースポリシーを使用して設定されます。X-Ray で必要なアクセス許可を使用する方法の詳細については、「Amazon SNS と AWS X-Ray」を参照してください。

Amazon SNS トピックでアクティブトレースを有効にする (コンソール)

Amazon SNS トピックでアクティブトレースが有効になると、トレース ID を読み取り、トレース ID に基づいてデータをカスタマーに送信し、そのトレース ID をダウンストリームのサービスに伝播します。

1. Amazon SNS コンソールにサインインします。
2. トピックを選択するか、新しいトピックを作成できます。トピックの作成の詳細については、「Amazon SNS のトピックの作成」を参照してください。
3. [トピックの作成] ページの [詳細] セクションで、[FIFO] または [標準] を選択します。
   a. トピックの名前を入力します。
   b. (オプション) トピックの表示名を入力します。
4. [Active tracing] (アクティブトレース) を展開し、[Use active tracing] (アクティブトレースを使用する) を選択します。

Amazon SNS トピックで X-Ray を有効にすると、X-Ray サービスマップを使用してトピックの end-to-end トレースとサービスマップを表示できます。

Amazon SNS トピックでアクティブトレースを有効にする (AWS SDK)

次のコード例は、AWS SDK for Java を使用して Amazon SNS トピックでアクティブトレースを有効にする方法を示しています。

```
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());
        }
    }
```

Amazon SNS トピックでアクティブトレースを有効にする (AWS CLI)

次のコード例は、AWS CLI を使用して Amazon SNS トピックでアクティブトレースを有効にする方法を示しています。

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

Amazon SNS トピックでアクティブトレースを有効にする (AWS CloudFormation)

次の AWS CloudFormation スタックは、Amazon SNS トピックでアクティブトレースを有効にする方法を示しています。

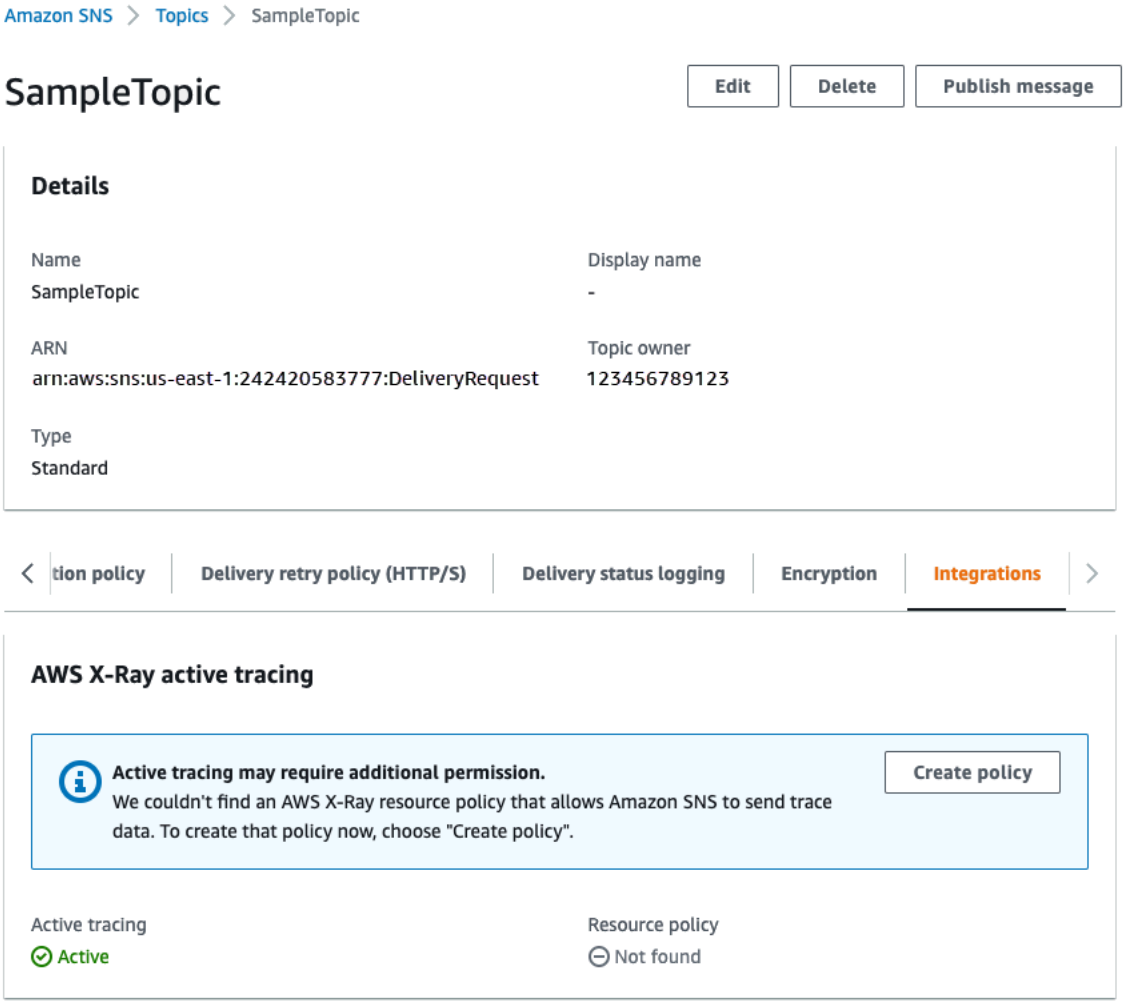
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

トピックでアクティブトレースが有効になっていることを確認する

Amazon SNS コンソールを使用して、トピックでアクティブトレースが有効になっているかどうか、またはリソースポリシーの追加に失敗したかどうかを確認できます。

1. Amazon SNS コンソールにサインインします。
2. 左のナビゲーションペインで、[トピック] を選択します。
3. [Topics] (トピック) ページで、トピックを選択します。
4. [Integrations] (統合) タブを開きます。

   アクティブトレースが有効になっている場合、[Active] (有効化) アイコンが表示されます。
5. アクティブトレースを有効にしているのにリソースポリシーが追加されていない場合は、[Create policy] (ポリシーの作成) を選択して必要なアクセス許可を追加します。



アクティブトレースのテスト

1. Amazon SNS コンソールにサインインします。
2. Amazon SNS トピックを作成します。この操作の詳細については、「を使用してトピックを作成するには AWS Management Console」を参照してください。
3. [Active tracing] (アクティブトレース) を展開し、[Use active tracing] (アクティブトレースを使用する) を選択します。
4. Amazon SNS トピックにメッセージを発行します。この操作の詳細については、「AWS Management Consoleを使用してAmazon SNS トピックにメッセージを発行するには」を参照してください。
5. X-Ray サービスマップを使用して、トピックの end-to-end トレースとサービスマップを表示します。


51ms
0.2 t/min
https://sqs.us-east-1...
AWS::SQS::Queue
11ms
0.2 t/min
xray-topic
AWS::SNS::Topic
77ms
0.2 t/min
xray-lambda
AWS::Lambda
Client
387ms
0.2 t/min
remote
13ms
0.2 t/min
KinesisFirehose
AWS::KinesisFirehose

ドキュメント履歴

以下の表は、「Amazon Simple Notification Service デベロッパーガイド」の最近の変更点をまとめています。

サービス機能は、 AWS サービスが利用可能なリージョンに段階的に展開されることがあります。このドキュメントは、最初のリリースのためにのみ更新されています。リージョンの可用性に関する情報を提供したり、その後のリージョンのロールアウトを発表したりすることはありません。サービス機能のリージョンでの提供状況や、更新に関する通知の受信登録方法については、「新着情報」を参照してください。 AWS。

変更	説明	日付
カナダ西部 (カルガリー) の FIFO トピックのサポート	Amazon SNS はカナダ西部 (カルガリー) の FIFO トピックをサポートしています。	2024 年 3 月 28 日
5 つの新しいリージョンでの Amazon SNS SMS サポート	Amazon SNS は、アジアパシフィック (ハイデラバード)、アジアパシフィック (メルボルン)、中東 (UAE)、ヨーロッパ (チューリッヒ)、ヨーロッパ (スペイン) の SMS サポートを追加しました。	2024 年 2 月 8 日
Firebase クラウドメッセージング (FCM) HTTP v1 サポート	Amazon SNS は FCM v1 の認証情報をサポートしています。	2024 年 1 月 18 日
アジアパシフィック (ジャカルタ) では、Amazon SNS は SMS をサポート	アジアパシフィック (ジャカルタ) では、Amazon SNS は SMS メッセージングをサポートしています。	2023 年 12 月 14 日
AWS CloudFormation Amazon SNS `DeliveryStatusLogging` トピックの設定のサポート	AWS CloudFormation Amazon SNS `DeliveryStatusLogging` トピックを作成または更	2023 年 12 月 7 日

	新する際の構成がサポートされています。	
新しいメッセージフィルタリング演算子が追加されました	Amazon SNS メッセージをフィルタリングするときに、サフィックスマッチング、equals-ignore case、および OR 演算子を使用できるようになりました。	2023 年 11 月 16 日
メッセージのアーカイブとリプレイに関するサポートが追加されました	トピック所有者はメッセージを最大 365 日間トピックにアーカイブできます。トピックサブスクライバーは、アーカイブされたメッセージをサブスクライブされたエンドポイントにリプレイして、ダウンストリームアプリケーションの障害によるメッセージを回復したり、既存のアプリケーションの状態を複製したりできます。	2023 年 10 月 26 日
標準キューを FIFO トピックにサブスクライブするためのサポートを追加	Amazon SQS FIFO キューまたは標準キューを Amazon SNS FIFO トピックにサブスクライブできます。Amazon SQS FIFO キューに限り、メッセージが順番に受信され、重複がないことが保証されます。	2023 年 9 月 14 日
イスラエル (テルアビブ) 向けの SMS サポートが追加されました	Amazon SNS SMS がイスラエル (テルアビブ) リージョンでサポートされるようになりました。	2023 年 8 月 28 日

FIFO トピックに追加された X-Ray アクティブトレースのサポート	以前は Amazon SNS AWS X-Ray 標準トピックでのみサポートされていましたが、FIFO トピックを通じて Amazon Data Firehose、Amazon SQS、 AWS Lambda および HTTP/S エンドポイントサブスクリプションに送信されるユーザーリクエストをトレースして分析するようになりました。	2023 年 5 月 31 日
Content-Type ヘッダーのサポート強化	リクエストポリシーに Content-Type ヘッダーを設定して、通知のメディアタイプを指定できます。	2023 年 3 月 23 日
アクティブトレースのサポートが追加されました	AWS X-Ray ユーザーリクエストが Amazon SNS 標準トピックを経由して Amazon Data Firehose、Amazon SQS、および HTTP/S エンドポイントサブスクリプションに送信されるときに AWS Lambda、ユーザーリクエストを追跡して分析します。	2023 年 2 月 8 日
シンガポールの送信者 ID 登録	シンガポールで送信者 ID を登録するための手順が追加されました。	2023 年 1 月 10 日
ペイロードベースのメッセージフィルタリング	ペイロードベースのフィルタリングにより、メッセージペイロードに基づいてメッセージをフィルタリングできるため、不要なデータの処理に伴うコストを回避できます。	2022 年 11 月 22 日

Amazon SNS メッセージ署名用に SHA256 ハッシュアルゴリズムが追加されました	Amazon SNS メッセージ署名を使用する際の SHA256 ハッシュアルゴリズムのサポートが追加されました。	2022 年 9 月 15 日
SMS メッセージングにリージョンが追加されました	Amazon SNS は、アフリカ (ケープタウン)、アジア太平洋 (大阪)、ヨーロッパ (ミラノ)、AWS GovCloud (米国東部) の SMS メッセージングをサポートしています。	2022 年 9 月 9 日
メッセージデータ保護サポートが追加されました	メッセージデータ保護は、AWS データ保護ポリシーを使用してアプリケーションまたはサービス間を移動する機密情報を監査およびブロックすることにより、Amazon SNS トピックに公開されたデータを保護します。	2022 年 9 月 8 日
通話料無料番号の新規登録プロセス	通話料無料の電話番号 (TFN) を使用して Amazon SNS メッセージを米国の受信者に送信するためのサポートを追加しました。	2022 年 8 月 1 日

属性ベースのアクセスコントロール (ABAC) のサポート	Publish および PublishBatch を含め、API アクションの属性ベースのアクセスコントロール (ABAC) のサポートが追加されました。ABAC は、アクセス権限管理を簡素化するために、IAMユーザーやロールなどのIAMリソースや、Amazon SNS AWS トピックなどのリソースにアタッチできるタグに基づいてアクセス権限を定義する認証戦略です。	2022 年 1 月 10 日
プッシュ通知用の Apple トークンベース認証のサポート	アプリのデベロッパーであることを識別する情報を提供することで、Amazon SNS に iOS または macOS アプリへのプッシュ通知の送信を許可できます。	2021 年 10 月 28 日
SMS メッセージの新しい送信者は SMS サンドボックスに配置される	SMS サンドボックスは、不正使用や悪用を防止し、送信者としての評判を保つのに役立ちます。 AWS アカウントが SMS サンドボックスにある間は、検証済みの宛先の電話番号にのみ SMS メッセージを送信できます。	2021 年 6 月 1 日

SMS メッセージの新しい送信者は SMS サンドボックスに配置される	SMS サンドボックスは、不正使用や悪用を防止し、送信者としての評判を保つのに役立ちます。 AWS アカウントが SMS サンドボックスにある間は、確認済みの宛先の電話番号にのみ SMS メッセージを送信できます。	2021 年 6 月 1 日
インドの受信者に SMS メッセージを送信する新しい属性	2 つの新しい属性、エンティティ ID およびテンプレート ID が、インドの受信者に SMS メッセージを送信するために必要になりました。	2021 年 4 月 22 日
メッセージフィルタリング演算子の更新	新しい演算子、`cidr` は、メッセージ送信元 IP アドレスとサブネットのマッチングに使用できます。また、属性キーがないかどうかをチェックし、属性文字列のマッチングのための `anything-but` 演算子でプレフィックスを使用します。	2021 年 4 月 7 日
米国の送信先の P2P ロングコードのサポート終了	2021 年 6 月 1 日をもって、米国の通信プロバイダーは、米国の宛先への person-to-person (A2P) 通信に (P2P) ロングコードを使用することをサポートしなくなりました。application-to-person 代わりに、短いコード、通話料無料の番号、または 10DLC と呼ばれる新しいタイプの発信番号を使用できます。	2021 年 2 月 16 日

1 分の Amazon CloudWatch メトリクスのSupport	Amazon SNS の 1 CloudWatch AWS 分間のメトリックスがすべてのリージョンで利用できるようになりました。	2021 年 1 月 28 日
Amazon データFirehose エンドポイントのSupport	Firehose 配信ストリームを SNS トピックに登録できます。これにより、Amazon Simple Storage Service (Amazon S3) バケット、Amazon Redshift テーブル、Amazon サービス (OpenSearch OpenSearch サービス) などのアーカイブおよび分析エンドポイントに通知を送信できます。	2021 年 1 月 12 日
送信元番号が利用可能	テキストメッセージ (SMS) の送信時に送信元番号を使用できます。	2020 年 10 月 23 日
Amazon SNS FIFO トピックのサポート	データの一貫性をほぼリアルタイムで必要とする分散アプリケーションを統合するには、Amazon SQS FIFO キューで Amazon SNS 先入れ先出し (FIFO) トピックを使用できます。	2020 年 10 月 22 日
Amazon SNS Extended Client Library for Java が利用可能	このライブラリを使用して、容量の大きい Amazon SNS メッセージを発行できます。	2020 年 8 月 25 日
SSE は、中国リージョンで使用可能	Amazon SNS のサーバー側の暗号化 (SSE) が、中国リージョンで利用可能です。	2020 年 1 月 20 日

DLQ を使用した配信不能メッセージのキャプチャのサポート	Amazon SQS デッドレターキュー (DLQ) を使用して、Amazon SNS サブスクリプションで配信不能メッセージをキャプチャできます。	2019 年 11 月 14 日
カスタム APN ヘッダーの値の指定のサポート	APN ヘッダーのカスタム値を指定できます。	2019 年 10 月 18 日
APN の 'apns-push-type ' ヘッダーフィールドのサポート	APN を介して送信されるモバイル通知に、apns-push-type ヘッダーフィールドを使用できます。	2019 年 9 月 10 日
を使用したトピックのトラブルシューティングのSupport AWS X-Ray	X-Ray を使用して、SNS トピックを通過するメッセージのトラブルシューティングを行うことができます。	2019 年 7 月 24 日
'exists' 演算子を使用した属性キーマッチングのサポート	フィルターポリシーにキーがリストされている属性が受信メッセージにあるかどうかをチェックするには、exists 演算子を使用できます。	2019 年 7 月 5 日
複数の数値の「以外」のマッチングのサポート	複数の文字列に加えて、Amazon SNS では複数の数値の「以外」のマッチングが可能です。	2019 年 7 月 5 日
Amazon SNS リリースノートが、RSS フィードとして利用できるようになりました	このページのタイトルに続いて ([ドキュメント履歴]) を選択し、[RSS] を選択します。	2019 年 6 月 22 日

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。