aws

開発者ガイド

Amazon Simple Queue Service

Amazon Simple Queue Service: 開発者ガイド

Table of Contents

Amazon Simple Queue Service とは

Amazon Simple Queue Service (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と分離ができる安全性、耐久性があり利用可能なホストキューを提供します。Amazon SQS は、デッドレターキューやコスト配分タグなど、一般的な構造を提供します。 AWS SDK がサポートする任意のプログラミング言語を使用してアクセスできる汎用ウェブサービス API を提供します。

トピック

Amazon SQSを使用する利点

- セキュリティ–Amazon SQS キューにメッセージを送信する人、およびメッセージを受信する人を制御します。Amazon SQS 管理のサーバー側の暗号化を使用するか、 AWS Key Management Service (AWS KMS) で管理されるカスタム SSE キーを使用して、キューのメッセージの内容を保護して、機密データを送信できます。
- 耐久性–メッセージの安全性のため、Amazon SQS は複数のサーバーにメッセージを保存します。スタンダードキューはat-least-once メッセージ配信 をサポートし、FIFO キューは厳密に 1 回限りのメッセージ処理と高スループットモードをサポートします。
- 可用性–Amazon SQS は、冗長なインフラストラクチャを使用して同時実行性が高いメッセージへのアクセスと、メッセージの作成と消費のための高い可用性を提供します。
- スケーラビリティ– Amazon SQSはバッファされたリクエストを独立して個別に処理できるため、プロビジョニング指示を必要とせずに負荷の増大や急増に対応できるよう透過的にスケーリングします。
- 信頼性–Amazon SQSは処理中にメッセージをロックするため、複数のプロデューサーがメッセージを送信し、複数のコンシューマーが同時にメッセージを受信できます。
- カスタマイズ-キューは完全に同じである必要はありません。たとえば、キューでデフォルトの遅延を設定できます。256 KB よりも大きいメッセージのコンテンツはAmazon Simple Storage Service (Amazon S3) または Amazon DynamoDBを使用し、Amazon SQSが、Amazon S3 オブジェクトへのポインタを保持して、保存できます。または、大きいメッセージを小さいメッセージに分割することができます。

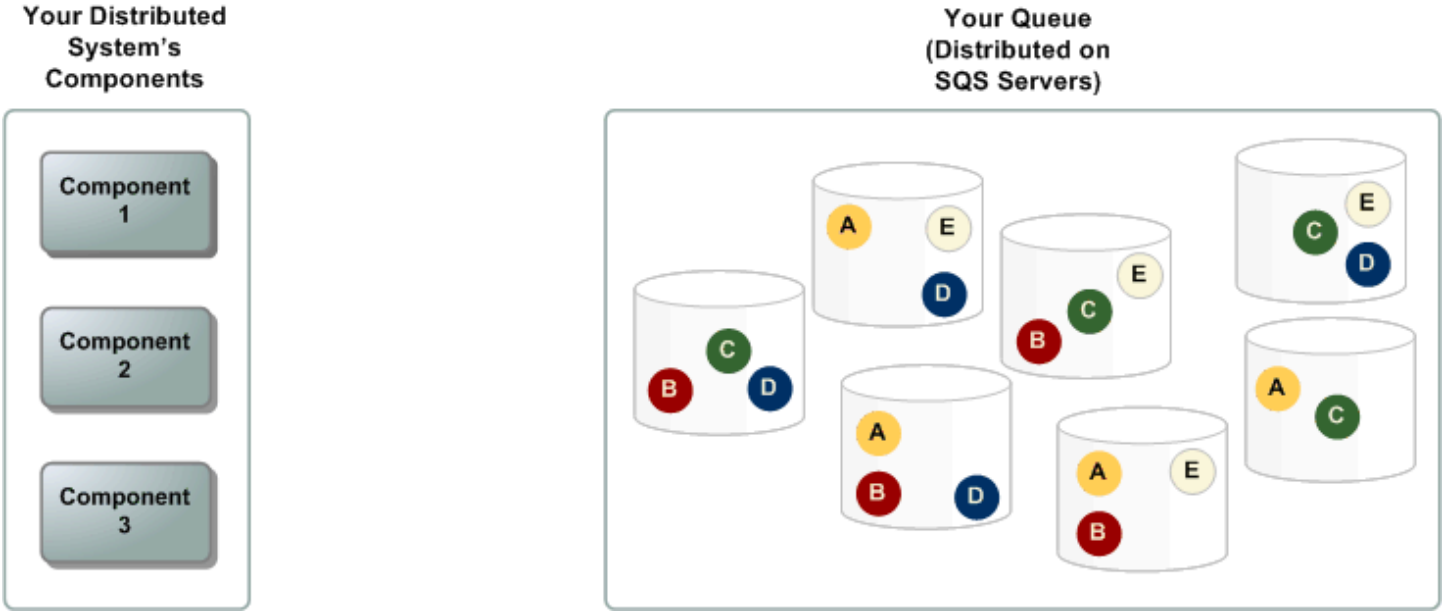
Amazon SQSの基本的なアーキテクチャ

このセクションでは、分散メッセージングシステムの各部分の概要およびAmazon SQSメッセージのライフサイクルについて説明します。

分散キュー

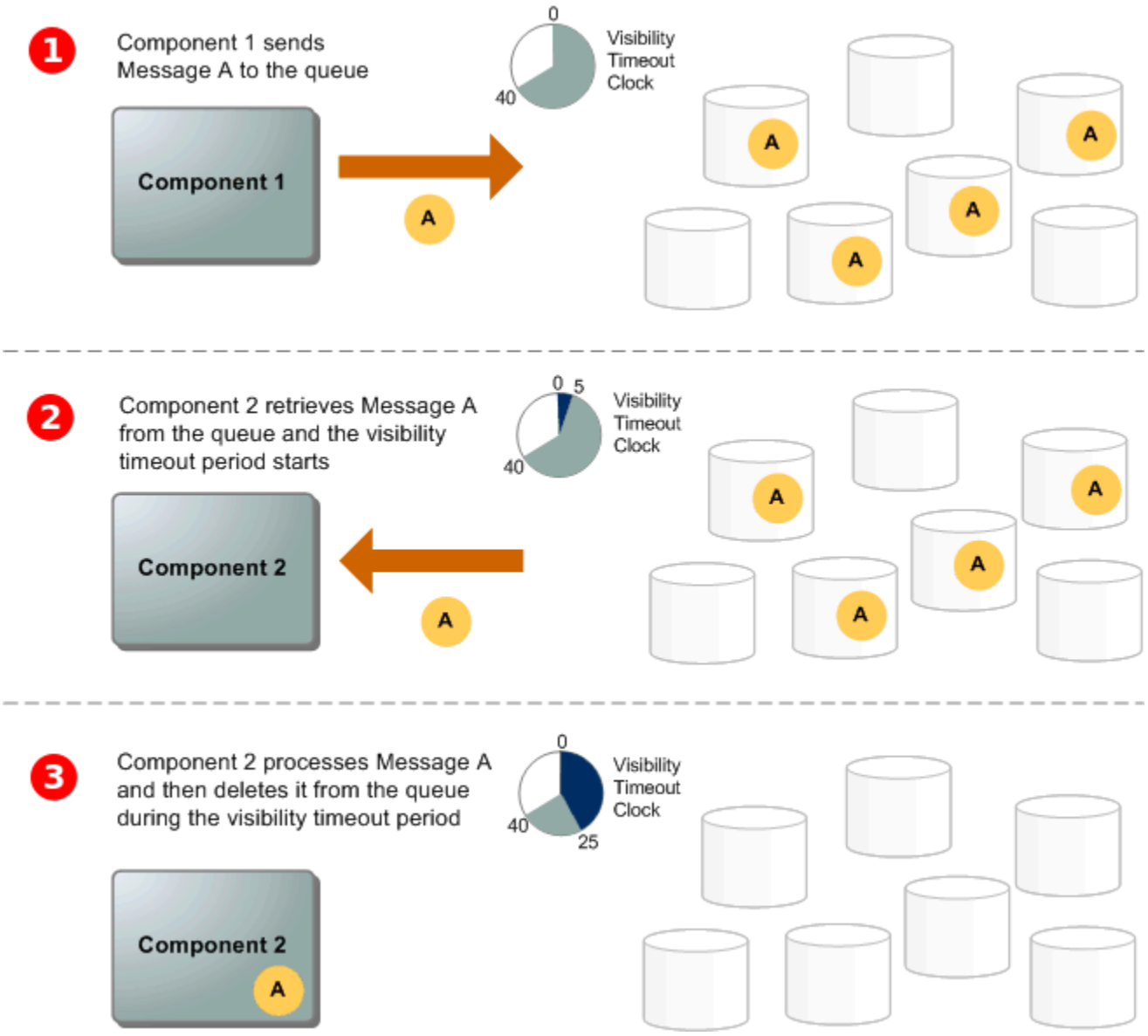
分散メッセージングシステムには 3 つの主要部分 (分散システムのコンポーネント、キュー (Amazon SQS サーバーで分散)、およびキューのメッセージ) があります。

次のシナリオでは、システムには複数のプロデューサー (キューにメッセージを送信するコンポーネント) とコンシューマー (キューからメッセージを受信するコンポーネント) があります。キュー (A から E までのメッセージを保持) は、複数のAmazon SQS サーバー全体にまたがって冗長的にメッセージを保存します。



メッセージのライフサイクル

次のシナリオは、キューのAmazon SQSメッセージのライフサイクルを作成から削除までを説明しています。



1

プロデューサー (コンポーネント 1) はメッセージ A をキューに送ります。このメッセージはAmazon SQSサーバー全体に重複して分散されます。

2

コンシューマー (コンポーネント 2) でメッセージを処理する準備が整うと、キューからメッセージを消費し、メッセージ A が返されます。メッセージ A は処理されている間キューに残り、可視性タイムアウトの間は次の受信リクエストに返されることはありません。

3

可視性タイムアウトの期限が切れると、コンシューマー (コンポーネント 2) はキューからメッセージ A を削除してメッセージが再び受信されて処理されるのを回避します。

> **Note**
>
> Amazon SQSは最大メッセージ保持期間を超えてキューに保存されたメッセージを自動的に削除します。デフォルトのメッセージ保持期間は 4 日間です。ただし、`SetQueueAttributes` アクションを使うと、メッセージ保持期間の値を 60 秒から 1,209,600 秒 (14 日間) までの範囲で設定できます。

Amazon SQS、Amazon MQとAmazon SNSの違い

Amazon SQS 、Amazon SNSは、高度にスケーラブルで easy-to-use管理されたメッセージングサービスを提供し、それぞれが分散システム内の特定のロール向けに設計されています。 Amazon MQ これらのサービスの違いに関する拡張された概要を次に示します。

Amazon SQS は、分散ソフトウェアシステムとコンポーネントをキューサービスとしてデカップリングしてスケーリングします。通常、1 人のサブスクライバーを通じてメッセージを処理し、順序と損失の防止が重要なワークフローに最適です。より広範なディストリビューションのために、Amazon SQS を Amazon SNS と統合すると、ファンアウトメッセージングパターン が可能になり、複数のサブスクライバーにメッセージを一度に効果的にプッシュできます。

Amazon SNS では、パブリッシャーは、通信チャネルとして機能するトピックを通じて複数のサブスクライバーにメッセージを送信できます。サブスクライバーは、、Amazon SQSAmazon Data Firehose、Lambda、HTTP、E メール、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メッセージを受信します。Amazon SQS このサービスは、リアルタイムのユーザーエンゲージメントやアラームシステムなど、即時の通知を必要とするシナリオに最適です。サブスクライバーがオフラインのときにメッセージが失われないように、Amazon SNS を Amazon SQS キューメッセージと統合することで、一貫した配信が保証されます。

Amazon MQ は、Apache ActiveMQ や RabbitMQ とともに AMQP や MQTT などの標準メッセージングプロトコルをサポートしており、従来のメッセージブローカーからの移行を検討している企業に最適です。 RabbitMQ 大幅な再設定を行わずに、安定した信頼性の高いメッセージングを必要とするレガシーシステムとの互換性を提供します。

次の表は、各サービスのリソースタイプの概要を示しています。

リソースタイプ	Amazon SNS	Amazon SQS	Amazon MQ
同期的	いいえ	いいえ	はい
非同期的	はい	はい	はい
キュー	いいえ	はい	はい
パブリッシャー/サブスクライバーメッセージング	はい	いいえ	はい
メッセージブローカー	いいえ	いいえ	はい

新規のアプリケーションには、Amazon SQS および Amazon SNS をお勧めします。ほぼ無制限のスケーラビリティとシンプルな API が利点です。通常、コスト効果の高いソリューションは、大量のアプリケーションに対してその pay-as-you-go 料金で提供されます。JMS などの APIs や、アドバンストメッセージキューイングプロトコル (AMQP)、MQTT、および Simple Text Oriented Message Protocol (STOMP) などのプロトコルとの互換性に依存する既存のメッセージブローカーからアプリケーションを移行するには OpenWire、Amazon MQ をお勧めします。

Amazon SQSのセットアップ

Amazon SQS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

任意の AWS サービスにアクセスするには、まずAWS アカウント、製品を使用できる Amazon.com アカウントである を作成する必要があります AWS 。を使用して AWS アカウント 、アクティビティと使用状況レポートを表示したり、認証とアクセスを管理したりできます。

Amazon SQS アクションに AWS アカウント ルートユーザーを使用しないようにするには、Amazon SQS への管理アクセスを必要とするユーザーごとに IAM ユーザーを作成することがベストプラクティスです。

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、次の手順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 アカウントの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。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

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、 を保護し AWS アカウントのルートユーザー、 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。

のセキュリティ保護 AWS アカウントのルートユーザー

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 IAM アイデンティティセンターディレクトリを設定する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というベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの参加」を参照してください。

ステップ 2: プログラム的なアクセス権を付与する

Amazon SQS アクションを使用するには (例えば、Java を使用するか、 を介して AWS Command Line Interface) 、アクセスキー ID とシークレットアクセスキーが必要です。

> **Note**
>
> アクセスキー ID とシークレットアクセスキーは に固有です AWS Identity and Access Management。Amazon EC2 キーペアなどの他の AWS サービスの認証情報と混同しないでください。

ユーザーが の AWS 外部で を操作する場合は、プログラムによるアクセスが必要です AWS Management Console。プログラムによるアクセスを許可する方法は、 にアクセスするユーザーのタイプによって異なります AWS。

ユーザーにプログラマチックアクセス権を付与するには、以下のいずれかのオプションを選択します。

プログラマチックアクセス権を必要とするユーザー	目的	方法
ワークフォースアイデンティティ (IAM Identity Center で管理されているユーザー)	一時的な認証情報を使用して、、 AWS SDKs AWS CLI、または AWS APIs。	使用するインターフェイス用の手引きに従ってください。 • については AWS CLI、「 ユーザーガイド」の AWS CLI 「 を使用するための の設定 AWS IAM Identity CenterAWS Command Line Interface 」を参照してください。 • AWS SDKs 、ツール、AWS APIs「 SDK とツールのリファレンスガイド」の「IAM Identity Center 認証」を参照してください。AWS SDKs
IAM	一時的な認証情報を使用して、、 AWS SDKs AWS CLI、または AWS APIs。	「IAM ユーザーガイド」の「 AWS リソースでの一時的な認証情報の使用」の手順に従います。
IAM	(非推奨) 長期認証情報を使用して、、 AWS SDKs AWS CLI、または AWS APIs。	使用するインターフェイス用の手引きに従ってください。 • については AWS CLI、「 AWS Command Line Interface ユーザーガイド」の「IAM ユーザー認証情報を使用した認証」を参照してください。 • AWS SDKs「 SDK とツールのリファレンスガイド」の「長期的な認証情報を使

プログラマチックアクセス権を必要とするユーザー	目的	方法
		用した認証」を参照してください。 AWS SDKs • AWS APIsユーザーガイド」の「IAM ユーザーのアクセスキーの管理」を参照してください。

ステップ 3: コード例を使用する準備を整える

このガイドには、 AWS SDK for Java を使用する例が含まれています。サンプルコードを実行するには、{AWS SDK for Java 2.0の使用開始 } セットアップ手順に従います。

Go、、Python、Ruby などJavaScript、他のプログラミング言語で AWS アプリケーションを開発できます。詳細については、「 で構築するツール AWS」を参照してください。

> **Note**
>
> (AWS CLI) や Windows などのツールを使用してコードを記述しなくても Amazon SQS を AWS Command Line Interface 試すことができます PowerShell。AWS CLI コマンドリファレンス の Amazon SQS セクションに AWS CLI 例があります。Windows PowerShell の例は、AWS Tools for PowerShell 「 コマンドレットリファレンス」の「Amazon Simple Queue Service」セクションで確認できます。

次のステップ

これで、「 AWS Management Consoleを使用してAmazon SQS キューとメッセージを管理する」を開始 する準備ができました。

Amazon SQSの開始方法

このセクションでは、Amazon SQS コンソールを使用して標準キューまたは FIFO キューを作成する方法について説明します。

トピック

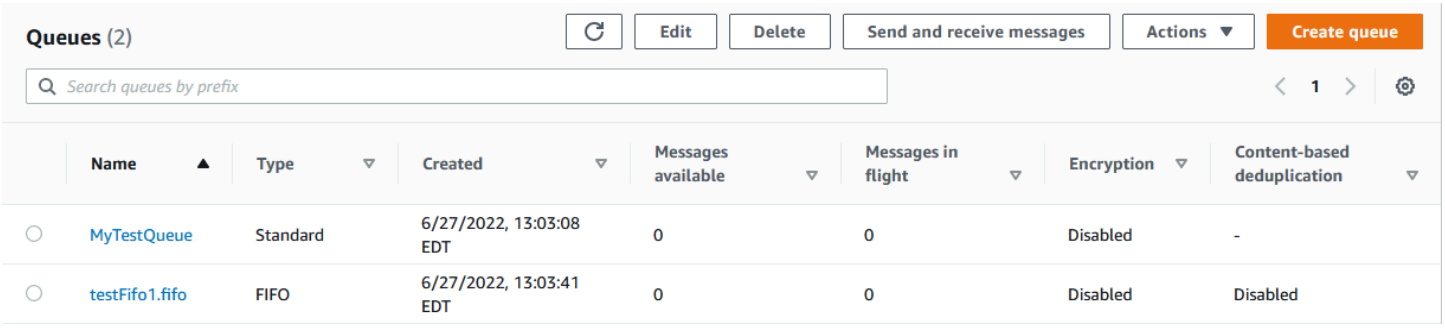
前提条件

開始する前に、「Amazon SQSのセットアップ」のステップを完了します。

Amazon SQS コンソールを理解する

Amazon SQS コンソールを開いたら、ナビゲーションペインからキューを選択します。{キュー}ページには、アクティブなリージョンのすべてのキューに関する情報が表示されます。

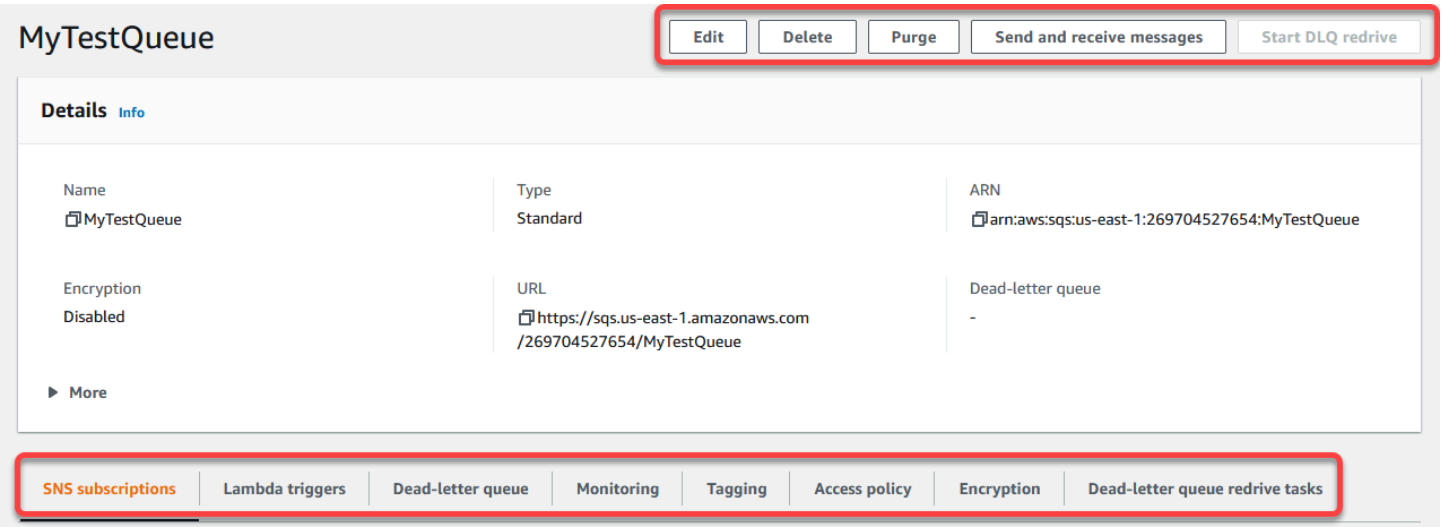
各キューエントリは、タイプやキー属性など、キューに関する重要な情報を提供します。最大スループットとベストエフォートメッセージの順序付けに最適化されたスタンダードキュー は、厳密なメッセージシーケンスを必要とするアプリケーションのメッセージの順序付けと一意性を優先する先入れ先出し (FIFO) キューと区別されます。



インタラクティブな要素とアクション

キューページには、キューを管理するための複数のオプションがあります。

1. クイックアクション – 各キュー名に隣接するドロップダウンメニューでは、メッセージの送信、メッセージの表示または削除、トリガーの設定、キュー自体の削除などの一般的なアクションにすばやくアクセスできます。
2. 詳細表示と設定 – キュー名をクリックすると、詳細ページが開き、キューの設定と設定をより深く掘り下げることができます。ここでは、メッセージ保持期間、可視性タイムアウト、最大メッセージサイズなどのパラメータを調整して、アプリケーションの要件に合わせてキューを調整できます。



リージョンの選択とリソースタグ

キューに効果的にアクセスして管理 AWS リージョン するために、正しい にいることを確認します。さらに、リソースタグを使用してキューを整理および分類し、 AWS共有環境内でリソース管理、コスト配分、アクセス制御を改善することを検討してください。

Amazon SQS コンソールで提供される機能を活用することで、メッセージングインフラストラクチャを効率的に管理し、キューのパフォーマンスを最適化し、アプリケーションに信頼性の高いメッセージ配信を確保できます。

Amazon SQS キュータイプ

Amazon SQS は、スタンダードキューと FIFO キューの 2 種類のキューをサポートしています。次の表の情報を使用して、状況に適したキューを選択してください。Amazon SQS キューの詳細については、「Amazon SQS スタンダードキューの開始方法」および「Amazon SQS での FIFO キューの開始方法」を参照してください。

スタンダードキュー	FIFO キュー
無制限スループット - スタンダードキューは、API アクションごと (SendMessage 、ReceiveMessage 、または DeleteMessage)、または 1 秒あたりほぼ無制限の API コールをサポートします。 少なくとも1回の配信:メッセージは少なくとも 1 回は確実に配信されますが、ときどき複数のメッセージのコピーが配信されることもあります。 ベストエフォート型の順序付け-ときどきメッセージが送信されたときと異なる順序で配信されることがあります。	高スループット—バッチ処理を使用した場合、FIFO キューは API メソッド (SendMessageBatch 、ReceiveMessage 、またはDeleteMessageBatch)ごとに 1 秒あたり最大 3,000通のメッセージをサポートします。1 秒あたり 3,000 通のメッセージは 300 回の API コールを表し、それぞれに 10 個のメッセージのバッチがあります。クォータの引き上げをリクエストするには、サポートリクエストを送信します。バッチ処理を行わない場合、FIFO キューは、APIメソッド (SendMessage 、ReceiveMessage 、または DeleteMessage) ごとに1秒あたり最大 300の API コールをサポートします。 1 回だけの処理 - メッセージは 1 度配信されると、ユーザーがそれを処理して削除するまでは使用可能な状態が保たれます。このキューでは、重複は導入されていません。 先入れ先出し配信-メッセージの送信または受信された順序は厳密に保たれます。
	
スループットを重視する場合、アプリケーション間でデータを送信します。例: • 負荷の高いバックグラウンドの作業からライブユーザーリクエストを分離します: ユーザーはサイズ変更またはエンコード中にメディアをアップロードできます。	イベントの順序を重視する場合、アプリケーション間でデータを送信します。例: • ユーザーが-入力したコマンドが正しい順序で実行されていることを確認します。 • 価格の変更を正しい順序で送信して、正しい製品価格を表示します。

スタンダードキュー	FIFO キュー
• 複数のワーカーノードにタスクを割り当てます:大量のクレジットカードの検証リクエストを処理します。 • 今後の処理のためのバッチメッセージ:データベースに追加する複数のエントリをスケジュールします。	• アカウントを登録する前に受講者がコースに登録できないようにします。

Amazon SQS スタンダードキューの作成とメッセージの送信

これは Amazon SQS のスタンダードキューを作成する方法です。

Amazon SQS コンソールを使用してキューを作成する

Amazon SQS コンソールを使用して、スタンダードキューを作成できます。コンソールには、キュー名以外のすべての設定にデフォルト値が表示されます。

> **Important**
>
> 2022 年 8 月 17 日に、Amazon SQS キューにデフォルトのサーバー側の暗号化が適用されました。
> 個人を特定できる情報 (PII) やその他の機密情報をキュー名に追加しないでください。キュー名には、請求や CloudWatch ログなど、多くの Amazon Web Services からアクセスできます。キュー名はプライベートまたは機密データに使用することを意図していません。

Amazon SQS スタンダードキューを作成する方法

1. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/。
2. [キューの作成]を選択します。
3. タイプを使用する場合、スタンダードキュータイプはデフォルトで設定されています。

> **Note**
>
> キューの作成後は、キューのタイプを変更することはできません。

4. キュー名を入力します。
5. （オプション）コンソールはキュー設定パラメータにデフォルト値を設定します。設定では、以下のパラメータに新しい値を設定することができます。

   a. 可視性タイムアウトを使用する場合、期間と単位を入力します。指定できる範囲は0秒から12時間です。デフォルト値は30秒です。

   b. メッセージの保持期間を使用する場合、期間と単位を入力します。指定できる範囲は1分から14日です。デフォルト値は4日です。

   c. 配信の遅れを使用する場合、期間と単位を入力します。指定できる範囲は0から15分です。デフォルト値は0秒です。

   d. 最大メッセージサイズを使用する場合、値を入力します。指定できる指定できる範囲は1KBから256KB です。デフォルト値は、256KBです。

   e. メッセージの受信待ち時間を使用する場合、値を入力します。指定できる範囲は0から20秒です。デフォルト値は0秒で、ショートポーリングを設定します。0以外の値を指定すると、ロングポーリングが設定されます。
6. (オプション)アクセスポリシーの定義 アクセスポリシーキューにアクセスできるアカウント、ユーザー、ロールを定義します。アクセスポリシーでは、ユーザーがアクセスできるアクションも定義します (SendMessage、ReceiveMessage、 またはDeleteMessageなど) 。デフォルトポリシーでは、キューの所有者のみがメッセージの送受信を許可されます。

   アクセスポリシーを定義するには、次のいずれかの操作を行います。

   - キューにメッセージを送信できるユーザーと、キューからメッセージを受信できるユーザーを設定するにはベーシックを選択します。コンソールは、選択に基づいてポリシーを作成し、結果のアクセスポリシーを読み取り専用 JSON パネルに表示します。
   - JSONアクセスポリシーを直接変更する場合は、アドバンストを選択します。これにより、各プリンシパル (アカウント、ユーザー、またはロール) が実行できるアクションのカスタムセットを指定できます。
7. リドライブ許可ポリシーでは、[有効] を選択します。[すべて許可]、[キュー別]、または [すべて拒否] のいずれかを選択します。キュー別選択する場合、Amazon リソースネーム (ARN) で、最大10個のソースキューのリストを指定します。
8. Amazon SQS では、デフォルトでサーバー側の暗号化が使用されます。暗号化キータイプを選択するか、Amazon SQS が管理するサーバー側の暗号化を無効にするには、[暗号化] を展開します。暗号化キータイプの詳細については、「SQS マネージド暗号化キーを使用したキューのサーバー側の暗号化の設定」および「Amazon SQS コンソールを使用したキューのサーバー側の暗号化の設定」を参照してください。

> **Note**
>
> SSE を有効にすると、暗号化されたキューへの匿名の SendMessage リクエストと ReceiveMessage リクエストは拒否されます。Amazon SQS のセキュリティベストプラクティスでは、匿名リクエストを使用しないことを推奨しています。Amazon SQS キューに匿名リクエストを送信する場合は、必ず SSE を無効にしてください。

9. (オプション) 配信不能メッセージを受信するデッドレターキューを設定するためには、デッドレターキューを拡大します。
10. (オプション)キューのタグを追加するには、タグを拡大します。
11. [キューの作成]を選択します。Amazon SQSはキューを作成し、キューの詳細ページが表示されます。

Amazon SQSは、新しいキューに関する情報をシステム全体に伝えます。Amazon SQS は分散システムであるため、コンソールがキューページにキューを表示するまでに多少の遅れが生じることがあります。

メッセージの送信

キューを作成したら、そのキューにメッセージを送信できます。

1. 左のナビゲーションペインで {キュー} を選択します。キューのリストから作成したキューを選択します。
2. アクションからメッセージの送信と受信を選択します。

   コンソールにメッセージの送信と受信ページが表示されます。
3. {メッセージ本文}に[メッセージテキスト] を (入力) します。
4. スタンダードキューの場合、[配信の遅延] の値を入力し、単位を選択します。例えば、60 と入力し、[秒]を選択します。詳細については、「Amazon SQSメッセージタイマー」を参照してください。
5. [メッセージの送信] を選択します。

   メッセージが送信されると、コンソールは成功メッセージを表示します。詳細を表示を選択して、送信されたメッセージに関する情報を表示します。

Amazon SQS FIFO キューを作成してメッセージを送信する

これは Amazon SQS 用の FIFO キューを作成する方法です。

キューを作成する

Amazon SQS コンソールを使用して、標準キューおよび FIFO キューを作成できます。コンソールには、キュー名以外のすべての設定にデフォルト値が表示されます。

> **Important**
>
> 2022 年 8 月 17 日に、Amazon SQS キューにデフォルトのサーバー側の暗号化が適用されました。
> 個人を特定できる情報 (PII) やその他の機密情報をキュー名に追加しないでください。キュー名には、請求や CloudWatch ログなど、多くの Amazon Web Services がアクセスできます。キュー名はプライベートまたは機密データに使用することを意図していません。

Amazon SQS FIFO キュー を作成する方法

1. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/。
2. [キューの作成]を選択します。
3. タイプを使用する場合、スタンダードキュータイプはデフォルトで設定されています。FIFO キューを作成するには、FIFOを選択します。

   > **Note**
   >
   > キューの作成後は、キューのタイプを変更することはできません。

4. キュー名を入力します。

   FIFOキューの名前は.fifoのサフィックスで終わる必要があります。サフィックスは80文字のキュー名クォータにカウントされます。キューがであるかどうかを確認するにはFIFOでは、キュー名の末尾がサフィックスで終わるかどうかでチェックすることがきます。

5. （オプション）コンソールはキュー設定パラメータにデフォルト値を設定します。設定では、以下のパラメータに新しい値を設定することができます。

a. 可視性タイムアウトを使用する場合、期間と単位を入力します。指定できる範囲は0秒から12時間です。デフォルト値は30秒です。

b. メッセージの保持期間を使用する場合、期間と単位を入力します。指定できる範囲は1分から14日です。デフォルト値は4日です。

c. 配信の遅れを使用する場合、期間と単位を入力します。指定できる範囲は0から15分です。デフォルト値は0秒です。

d. 最大メッセージサイズを使用する場合、値を入力します。指定できる指定できる範囲は1KBから256KB です。デフォルト値は、256KBです。

e. メッセージの受信待ち時間を使用する場合、値を入力します。指定できる範囲は0から20秒です。デフォルト値は0秒で、ショートポーリングを設定します。0以外の値を指定すると、ロングポーリングが設定されます。

f. FIFO キューの場合は、コンテンツベースの重複除外を有効にするために [コンテンツベースの重複除外] を選択します。デフォルト設定は無効です。

g. (オプション) FIFO キューの場合、キュー内のメッセージの送受信でより高スループットを有効にするには、[高スループット FIFO を有効にする] を選択します。

このオプションを選択すると、関連するオプション (重複除外のスコープおよびFIFO スループットの制限) を使用して、FIFO キューの高スループットを有効にするために必要な設定に変更されます。高スループット FIFO の使用に必要な設定のいずれかを変更すると、キューに対して通常のスループットが有効になり、指定されたとおりに重複除外が実行されます。詳細については、「Amazon SQS の FIFO キューの高スループット」および「Amazon SQS メッセージクォータ」を参照してください。

6. (オプション)アクセスポリシーの定義 アクセスポリシーキューにアクセスできるアカウント、ユーザー、ロールを定義します。アクセスポリシーでは、ユーザーがアクセスできるアクションも定義します (SendMessage、ReceiveMessage、 またはDeleteMessageなど) 。デフォルトポリシーでは、キューの所有者のみがメッセージの送受信を許可されます。

アクセスポリシーを定義するには、次のいずれかの操作を行います。

- キューにメッセージを送信できるユーザーと、キューからメッセージを受信できるユーザーを設定するにはベーシックを選択します。コンソールは、選択に基づいてポリシーを作成し、結果のアクセスポリシーを読み取り専用 JSON パネルに表示します。
- JSONアクセスポリシーを直接変更する場合は、アドバンストを選択します。これにより、各プリンシパル (アカウント、ユーザー、またはロール) が実行できるアクションのカスタムセットを指定できます。

7. リドライブ許可ポリシーでは、[有効] を選択します。[すべて許可]、[キュー別]、または [すべて拒否] のいずれかを選択します。キュー別選択する場合、Amazon リソースネーム (ARN) で、最大10個のソースキューのリストを指定します。
8. Amazon SQS では、デフォルトでサーバー側の暗号化が使用されます。暗号化キータイプを選択するか、Amazon SQS が管理するサーバー側の暗号化を無効にするには、[暗号化] を展開します。暗号化キータイプの詳細については、「SQS マネージド暗号化キーを使用したキューのサーバー側の暗号化の設定」および「Amazon SQS コンソールを使用したキューのサーバー側の暗号化の設定」を参照してください。

   > **Note**
   >
   > SSE を有効にすると、暗号化されたキューへの匿名の SendMessage リクエストと ReceiveMessage リクエストは拒否されます。Amazon SQS のセキュリティベストプラクティスでは、匿名リクエストを使用しないことを推奨しています。Amazon SQS キューに匿名リクエストを送信する場合は、必ず SSE を無効にしてください。

9. (オプション) 配信不能メッセージを受信するデッドレターキューを設定するためには、デッドレターキューを拡大します。
10. (オプション)キューのタグを追加するには、タグを拡大します。
11. [キューの作成]を選択します。Amazon SQSはキューを作成し、キューの詳細ページが表示されます。

Amazon SQSは、新しいキューに関する情報をシステム全体に伝えます。Amazon SQS は分散システムであるため、コンソールがキューページにキューを表示するまでに多少の遅れが生じることがあります。

キューを作成したら、メッセージの送信およびメッセージの受信と削除ができます。また、キュータイプ以外のキュー構成の設定も編集することができます。

メッセージの送信

キューを作成したら、そのキューにメッセージを送信できます。

1. 左のナビゲーションペインで {キュー} を選択します。キューのリストから作成したキューを選択します。
2. アクションからメッセージの送信と受信を選択します。

   コンソールにメッセージの送信と受信ページが表示されます。

3. {メッセージ本文}に[メッセージテキスト] を (入力) します。
4. 先入れ先出し (FIFO) キューの場合、{ メッセージグループ ID } を (入力) します。詳細については、「Amazon SQS の FIFO キュー配信口ジック」を参照してください。
5. (オプション) FIFO キューの場合、メッセージ重複除外IDを入力できます。キューでコンテンツベースの重複除外を有効にした場合、メッセージ重複除外IDは必要ありません。詳細については、「Amazon SQS の FIFO キュー配信口ジック」を参照してください。
6. FIFO キューは、個々のメッセージのタイマーをサポートしていません。詳細については、「Amazon SQSメッセージタイマー」を参照してください。
7. [メッセージの送信] を選択します。

   メッセージが送信されると、コンソールは成功メッセージを表示します。詳細を表示を選択して、送信されたメッセージに関する情報を表示します。

Amazon SQS キューの管理

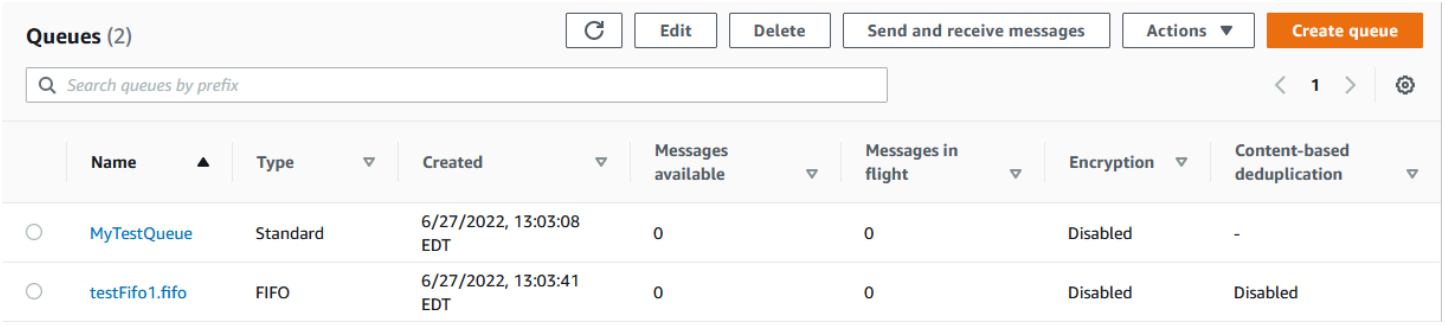
このセクションは、 Amazon SQSコンソール を使用してキューおよびメッセージを管理方法を示すことで Amazon SQSの理解をさらに深めるために役立ちます。

前提条件

開始する前に、「Amazon SQSのセットアップ」のステップを完了します。

Amazon SQS コンソールを理解する

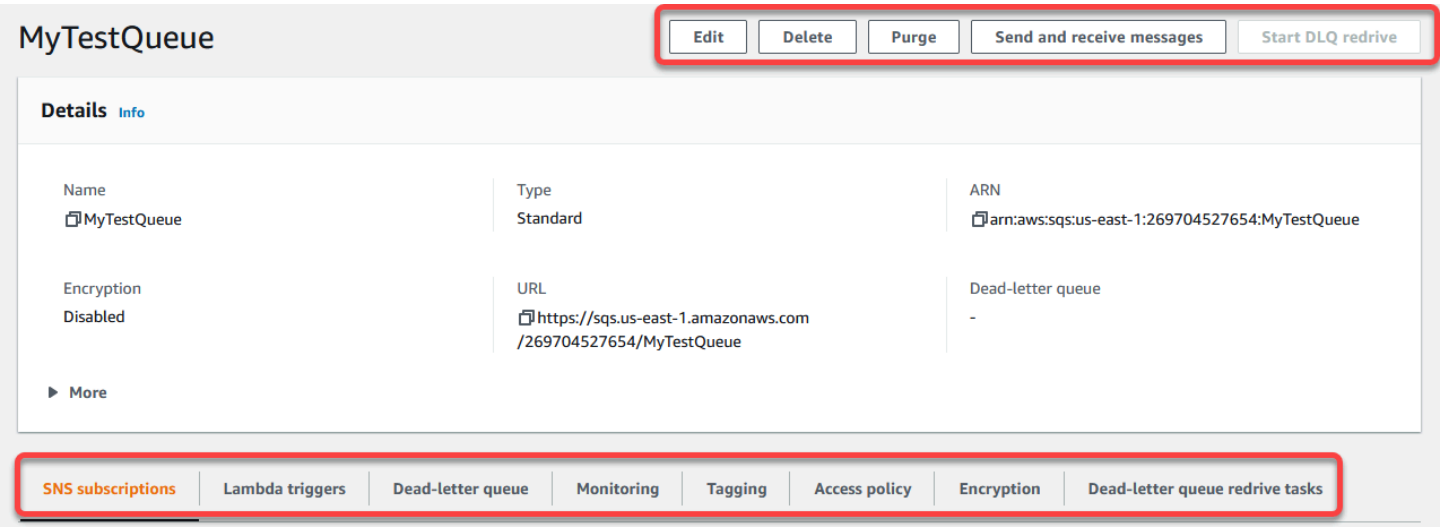
コンソールを開く時、{キュー}ページを表示するためにナビゲーションペインから [キュー]を選びます。{キュー}ページには、アクティブなリージョンのすべてのキューに関する情報が表示されます。



各キューのエントリには、キューのタイプとキューに関するその他の情報が表示されます。そのタイプ列は標準キューと先入れ先出し (FIFO) キューから一見しただけで区別するのに役立ちます。

その{キュー}ページから、キューに対してアクションを実行する方法は 2 つあります。キューの名の横にあるオプションを選択し、そのキューについて実行したいアクションを選択できます。

また、キュー名を選択することもできます。これにより、キューの詳細ページが表示されます。この詳細ページには、{キュー}ページとして同じアクションが含まれています。さらに、詳細の下ある次のタブのいずれかを選択すると、追加の構成詳細 とアクションを表示できます。



コンソールを使用した Amazon SQS キューの編集

Amazon SQSコンソールを使用して、キュー設定パラメータ (キュータイプを除く) を編集し、キュー機能を追加または削除できます。

Amazon SQSキューを編集するには(コンソール)

1. Amazon SQSコンソールの キュー ページを開きます。
2. キューを選択し、[編集] を選択します。
3. (オプション)設定で、キュー設定パラメータを更新する。
4. (オプション)アクセスポリシー を更新するには、アクセスポリシーで、JSONポリシーを変更します。
5. (オプション) デッドレターキューのリドライブ許可ポリシーを更新するには、[リドライブ許可ポリシー] を展開します。
6. (オプション) 暗号化を更新または削除するには、[暗号化] を展開します。
7. (オプション)デッドレターキュー (配信不能メッセージを受信できる) を追加、更新、または削除するには、デッドレターキュー拡大します。
8. (オプション)タグキューを追加、更新、または削除するには、タグを拡大します。
9. [Save]を選択します。

   コンソールはキューの詳細ページを表示します。

Amazon SQS でのメッセージの受信と削除

Amazon SQS キューにメッセージを送信した後、メッセージを受信および削除するオプションがあります。キューからメッセージをリクエストする場合、個々のメッセージを指定することはできません。代わりに、取得するメッセージの最大数を 10 個まで決定します。

Amazon SQS は分散システムとして機能し、メッセージが少ないキューからメッセージを取得すると、空のレスポンスが発生することがあります。その場合は、リクエストを再実行してください。メッセージの取得を最適化し、空のレスポンスを最小限に抑えるには、ロングポーリング の使用を検討してください。ロングポーリングは、メッセージが使用可能になるか、ポーリングがタイムアウトするまで応答を遅らせ、不要なポーリングコストを削減し、効率を向上させます。

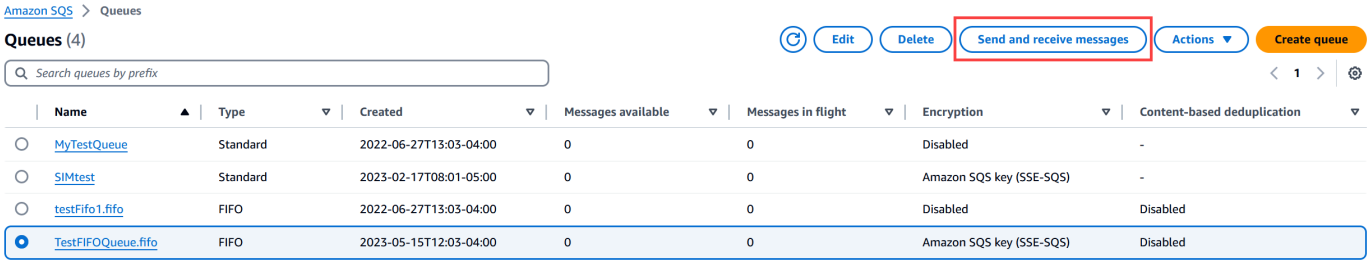
Amazon SQS では、アプリケーションやネットワークの中断などの処理の失敗によってメッセージにアクセスできなくなることがないようにするため、メッセージは取得後に自動的に削除されません。キューからメッセージを完全に削除するには、メッセージの処理後に削除リクエストを明示的に送信して、受信と処理が正常に完了したことを確認する必要があります。

Amazon SQS コンソールを介してメッセージが取得されると、再取得のためにすぐに再び表示されます。このデフォルトの動作により、手動操作中にメッセージが誤って失われることはなくなりますが、処理が繰り返される可能性があります。自動環境では、可視性タイムアウト設定を調整して、メッセージが取得された後も他のコンシューマーに見えないままになる時間を制御します。この設定は、複数のコンシューマー間でメッセージ処理を調整し、メッセージが 1 回だけ処理されるようにするために重要です。

メッセージの受信と削除の詳細なオペレーションについては、Amazon SQS API リファレンスガイド」を参照してください。このガイドでは、複雑なメッセージ処理シナリオを効果的に管理するパラメータなど、API エンドポイントに関する包括的な情報を提供します。

コンソールを使用してメッセージを受信および削除するには

1. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)]を選択します。
3. キューページでキュー を選択し、メッセージの送受信 を選択します。



4. メッセージの送信と受信ページで、メッセージのポーリング を選択します。

   Amazon SQSは、キューのメッセージのポーリングを開始します。メッセージの受信セクションの右側にあるプログレスバーには、ポーリングの期間が表示されます。

   メッセージセクションには、受信したメッセージの一覧表が表示されます。メッセージごとに、メッセージID、送信日、サイズ、および受信数の一覧表が表示されます。
5. メッセージを削除するには、削除するメッセージを選択し、削除を選択します。
6. メッセージの削除 ダイアログボックスで、 の削除 を選択します。

Amazon SQS キューが空であることを確認する

ほとんどの場合、キューが空かどうかを判断するためにロングポーリングが使用できます。まれに、キューにまだメッセージが含まれている場合でも、空の返信が受信されることがあります。特に作成したキューにメッセージの受信待ち時間を低い値を指定した場合にこのようなことがある場合があります。このセクションでは、キューが空であることを確認する方法について説明します。

キューが空であることを確認するには(コンソール)

1. すべてのプロデューサーのメッセージの送信を停止する。
2. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/ 。
3. ナビゲーションペインで [Queues(キュー)]を選択します。
4. キュー ページで、キュー を選択します。
5. モニタリングタブを選択します。
6. モニタリングダッシュボードの右上にある、{更新}マークの横にある下向き矢印を選択します。ドロップダウンメニューから[自動更新]を選択します。更新間隔は1分のままにしてください。
7. 次のダッシュボードを確認してください:

   - 遅延したメッセージの概数
   - 表示されないメッセージの概数

- 表示されるメッセージの概数

数分間、全ての0値が表示されるとキューは空になります。

キューが空であることを確認するには (AWS CLI， AWS API)

1. すべてのプロデューサーにメッセージの送信を停止。
2. 以下のいずれかのコマンドを繰り返し実行します。
   - AWS CLI: `get-queue-attributes`
   - AWS API: `GetQueueAttributes`
3. 以下の属性のメトリックスを確認してください。
   - `ApproximateNumberOfMessagesDelayed`
   - `ApproximateNumberOfMessagesNotVisible`
   - `ApproximateNumberOfMessagesVisible`

   それら全てが0数分間表示されると、キューは空になります。

Amazon CloudWatch メトリクスに依存している場合は、キューを空と見なす前に、複数のゼロデータポイントが連続して表示されることを確認してください。 CloudWatch メトリクスの詳細については、「」を参照してくださいAmazon SQS で使用できる CloudWatch メトリクス。

Amazon SQSキューの削除

Amazon SQSキューを使用しなくなり、近い将来使用が予測されない場合は、削除することをお勧めします。

> **Tip**
>
> キューを削除する前にキューが空であることを確認する場合は、Amazon SQS キューが空であることを確認するを参照してください

空でない場合でも、キューを削除できます。キュー内のメッセージを削除し、キュー自体を削除しない場合は、キューを消去することができます。

キューを削除するには(コンソール)

1. Amazon SQSコンソールを開きますhttps://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)]を選択します。
3. キューのページで{キューの削除}を選択します。
4. [削除] を選択します。
5. キューの削除ダイアログボックスで**delete**を入力して削除を確定します。
6. [削除] を選択します。

キュー (AWS CLI および API) を削除するには

キューを削除するには、以下のいずれかのコマンドを使用します。

- AWS CLI: `aws sqs delete-queue`
- AWS API: `DeleteQueue`

Amazon SQS コンソールを使用したキューからのメッセージの消去

Amazon SQSキューを削除せずに、キュー内のメッセージをすべて削除する必要がある場合は、キューを消去することができます。メッセージ削除プロセスには最大60秒かかります。キューのサイズにかかわらず、60秒待機することをお勧めします。

> **Important**
>
> キューを消去すると、削除されたメッセージを取得できなくなります。

キューを消去するには(コンソール)

1. Amazon SQSコンソールを開きますhttps://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues (キュー)]を選択します。
3. [キュー]ページで、消去するキューを選択します。
4. アクション から、 を消去 を選択します。

5. キューの消去ダイアログボックスで、次のように入力して消去を確認します。消去**purge**を選択します。

   すべてのメッセージがキューから消去されます。コンソールに確認バナーが表示されます。

Amazon SQSの使用開始の汎用タスク

キューを作成し、メッセージの送信、受信、削除方法およびキューの削除方法を理解したら、次のような設定を行うこともできます:

- Lambda 関数をトリガーする方法については、「AWS Lambda 関数をトリガーするための Amazon SQS キューの設定」を参照してください。
- SSE およびその他の機能を含むキューを設定する方法について説明します。
- 属性を含むメッセージを送信する方法について説明します。
- VPC からメッセージを送信する方法について説明します。
- Amazon SQS の機能とアーキテクチャについて知るためには、「Amazon SQS キュータイプ」および「Amazon SQSの基本的なアーキテクチャ」を参照してください。
- Amazon SQS を最大限に活用するためのガイドラインと注意事項については、{ Amazon SQS のベストプラクティス } を参照してください。
- AWS SDK for Java 2.x デベロッパーガイド など、 AWS SDKsつの Amazon SQS の例をご覧ください。
- Amazon SQS AWS CLI コマンドの詳細については、AWS CLI 「 コマンドリファレンス」を参照してください。
- Amazon SQS アクションの詳細については、Amazon Simple Queue Service API リファレンスを参照してください。
- Amazon SQS をプログラムで操作する方法について説明します。 APIs の使用を読み、AWS 開発センターをご覧ください。
  - Java
  - JavaScript
  - PHP
  - Python
  - Ruby
  - Windows & .NET

- 「Amazon SQS の問題のトラブルシューティング」セクションで、コストとリソースの管理方法について参照してください。
- 「セキュリティ」セクションで、データとデータへのアクセスの保護について参照してください。
- Amazon SQS ワークフローの詳細については、Amazon SQS アクセスコントロールプロセスワークフロー」セクションを参照してください。

Amazon SQS スタンダードキューの開始方法

Amazon SQSは、デフォルトのキュータイプとして「スタンダード」を提供します。スタンダードキューは、API アクション (SendMessage、ReceiveMessage、または DeleteMessage) ごとに 1 秒あたりほぼ無制限の API コールをサポートします。スタンダードキューは at-least-once メッセージ配信をサポートします。ただし、ときとして (ほぼ無制限のスループットが可能な高分散アーキテクチャであるために) メッセージの複数のコピーが順不同で配信されることがあります。スタンダードキューは、全般的にメッセージが送信順に配信されるベストエフォートタイプの順序を提供します。

Amazon SQS は、SendMessage が確認される前にメッセージを複数のアベイラビリティーゾーン (AZ) に冗長的に保存します。メッセージのコピーは複数の AZ に保存されるため、1 台のコンピュータ、ネットワークの 1 つ、または AZ の 1 つに障害が発生してもメッセージにアクセスできなくなることはありません。

Amazon SQS コンソールを使用してキューを作成および設定する方法については、{Amazon SQS コンソールを使用してキューを作成する}を参照してください。Javaの例については、Amazon SQS Java SDKの例を参照してください。

アプリケーションが、順不同で複数回到達するメッセージを処理できる限りは、多くのシナリオでスタンダード メッセージキューを使用できます。

- 負荷の高いバックグラウンドの作業からライブユーザーリクエストを分離します-ユーザーはサイズ変更またはエンコード中にメディアをアップロードできます。
- 複数のワーカーノードにタスクを割り当てます-大量のクレジットカードの検証リクエストを処理します。
- 今後の処理のためにメッセージをバッチ処理します -データベースへの複数のエントリの追加をスケジュールします。

スタンダードキューに関連するクォータについては、「クォータ」を参照してください。

スタンダードキューを使用した作業のベストプラクティスについては、「Amazon SQS 標準およびFIFOキューに関する推奨事項」を参照してください。

メッセージの順序

スタンダードキューでは、できる限りメッセージの順序を保持しますが、複数のメッセージのコピーが順不同で配信される場合があります。ご使用のシステムで順序の保持が必須の場合は、FIFOキュー (先入れ先出し) を使用するか、各メッセージに順序付け情報を追加して、受信時にメッセージを並べ直せるようにすることをお勧めします。

t-least-once 配信

Amazon SQSでは、冗長性と高可用性を確保するため、メッセージのコピーが複数のサーバーに保存されます。まれではありますが、メッセージを受信または削除するときに、メッセージのコピーが保存されているサーバーの1台が使用できない場合があります。

この場合、使用できないサーバーではメッセージのコピーが削除されず、メッセージを受信したときにそのメッセージのコピーが再度取得される可能性があります。アプリケーションがべき等になるよう設計する必要があります (同じメッセージを繰り返し処理した場合にも悪影響が発生しないように設計する必要があります)。

Amazon SQSキューとメッセージの識別子

このセクションでは、スタンダードキューおよびFIFOキューの識別子について説明します。これらの識別子は、特定のキューとメッセージを見つけて操作するうえで役立ちます。

Amazon SQS スタンダードキューの識別子

次の識別子の詳細については、Amazon Simpleキューサービス APIリファレンス。を参照してください

キュー名およびURL

新しいキューを作成する際は、 AWS アカウントおよびリージョンに一意なキュー名を指定する必要があります。Amazon SQSは、作成した各キューをキュー URLこれには、キュー名と他のAmazon SQSコンポーネントが含まれます。キューでアクションを実行するときは必ず、そのキュー URLを指定します。

次に示すのは、AWSアカウント番号MyQueueを持つユーザーにより所有される123456789012という名前のキューのキュー URL です。

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue
```

キューを一覧表示し、アカウント番号に続く文字列を解析することで、キューのURLをプログラムで取得できます。詳細については、「ListQueues」を参照してください。

メッセージ ID

各メッセージは、システムにより割り当てられたメッセージ ID を受け取り、Amazon SQS はSendMessage レスポンスにより返信します。この識別子は、メッセージを特定する場合に役立ちます。メッセージ IDの最大長は100文字です。

受信ハンドル

キューからメッセージを受信するたびに、そのメッセージの受信ハンドルを受け取ります。このハンドルは、メッセージ自体ではなくメッセージ受信のアクションと関連付けられます。メッセージを削除したり、メッセージ可視性を変更したりするには、受信ハンドル (メッセージ ID ではなく) を指定する必要があります。つまり、メッセージを削除する前にメッセージを受信する必要があります (メッセージをキューにおいてから回収することはできません)。受信ハンドルの最大長は1,024文字です。

> **Important**
>
> メッセージを複数回受信した場合、受信するたびに異なる受信ハンドルを受け取ります。メッセージの削除をリクエストするときは、最後に受け取った受信ハンドルを指定してください (そうしないと、メッセージが削除されない可能性があります)。

受信ハンドルの例を次に示します (3行に分割されています)。

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

クォータ

次の表に、スタンダードキューに関連するクォータの一覧を示します。

クォータ	説明
遅延キュー	キューのデフォルト (最小) 遅延は0秒です。最大は15分です。

クォータ	説明
表示されたキュー	`ListQueues` リクエストあたり1,000キュー。
ロングポーリングの待ち時間	ロングポーリングの最大待機時間は20秒です。
キューあたりのメッセージ (バックログ)	Amazon SQSキューに保存できるメッセージ数は無制限です。
キューあたりのメッセージ (処理中)	ほとんどのスタンダードキューでは (キュートラフィックとメッセージバックログにより異なります)、最大約120,000 通の処理中のメッセージ (コンシューマーからキューを受信し、キューからまだ削除されていないもの) が存在する可能性があります。ショートポーリングを使用している場合、このクォータに達すると、Amazon SQS は`OverLimit` エラーメッセージをし返信します。ロングポーリングを使用している場合、 Amazon SQSはエラーメッセージを返しません。クォータに到達しないように、処理されたメッセージはキューから削除する必要があります。メッセージの処理に使用するキューの数を増やすこともできます。クォータの引き上げをリクエストするには、サポートリクエストを送信します。
キュー名	キュー名には最大80文字を使用できます。次の文字を使用できます:英数字、ハイフン (-)、およびアンダースコア (_)。 **Note** キューの名前では、大文字と小文字が区別されます (たとえば、`Test-queue` と`test-queue` は異なるキューです)。
キューのタグ	50個を超えるタグをキューに追加することはお勧めしません。タグは、UTF-8 の Unicode 文字をサポートします。

クォータ	説明
	タグKeyは必須ですが、タグValueはオプションです。
	タグKeyとタグValueの大文字と小文字は区別されます。
	タグKeyとタグValueには、UTF-8のUnicode 英数字と空白を含めることができます。次の特殊文字を使用できます。_ . : / = + - @
	タグKeyまたはタグValueには、予約済みプレフィックスのaws:を含めることはできません (このプレフィックスが付いているタグキーやタグ値は削除できません)。
	タグKeyの最大長はUTF-8で128Unicode文字です。タグKeyを空またはnullにすることはできません。
	タグValueの最大長はUTF-8で256Unicode文字です。タグValueを空またはnullにすることはできません。
	タグ付けアクションは、 あたり 30 TPS に制限されています AWS アカウント。アプリケーションでより高いスループットが必要な場合、リクエストを送信します。。

Amazon SQS での FIFO キューの開始方法

FIFO (First-In-First-Out)キューは、スタンダードキューの全性能に加えて、オペレーションやイベントの順序が重要である場合、または重複不可の場合などにアプリケーション間のメッセージングを強化をするために設計されています。

FIFO キューを使用する可能性のある状況の例を次に示します。

1. 順序が重要な e コマース注文管理システム
2. イベントを順番に処理する必要があるサードパーティシステムとの統合
3. ユーザーが入力した入力を入力順に処理する
4. 通信とネットワーク — データと情報を同じ順序で送受信する
5. コンピュータシステム - ユーザーが入力したコマンドが正しい順序で実行されるようにする
6. 教育機関 - アカウントに登録前に受講者がコースに登録するのを防ぐ
7. オンラインチケットシステム — チケットを先着順で配布する

> **Note**
>
> また、FIFO キューはただ 1 回のみの処理を行いますが、1 秒あたりのトランザクション (TPS) 数は制限されます。FIFO キューで Amazon SQS 高スループットモードを使用すると、トランザクションの上限を引き上げることができます。高スループットモードの使用方法の詳細については、「Amazon SQS の FIFO キューの高スループット」を参照してください。スループットクォータの詳細については、「the section called “メッセージのクォータ”」を参照してください。

Amazon SQS FIFO キューは、Amazon SQSが利用可能なすべてのリージョンで利用可能です。

複雑な順序で FIFO キューを使用する方法の詳細については、「Amazon SQS FIFO キューによる複雑な順序付けの課題の解決」を参照してください。

Amazon SQS コンソールを使用してキューを作成および設定する方法については、{Amazon SQS コンソールを使用してキューを作成する}を参照してください。Javaの例については、{Amazon SQS Java SDKの例}を参照してください。

FIFOキューを使用した作業のベストプラクティスについては、{Amazon SQS FIFOキューのその他の推奨事項}および{Amazon SQS 標準および FIFOキューに関する推奨事項}を参照してください。

Amazon SQS の FIFO キュー配信ロジック

次の概念は、 FIFOから送信されるメッセージと受信されるメッセージをよりよく理解するうえで役立ちます。

メッセージの送信

複数のメッセージが連続してFIFOキューに送信され、それぞれに個別のメッセージ重複除外IDが付いている場合、 Amazon SQSはメッセージを保存し送信を認識します。その後、各メッセージを、送信された正確な順序で受信し処理できます。

FIFOキューでは、メッセージはメッセージグループ IDに基づいて順序付けられます。複数のホスト (または同一ホストの異なるスレッド) から同じメッセージグループ IDのメッセージがFIFOキューに送信された場合、Amazon SQSはメッセージを受信順に保存して処理を待ちます。Amazon SQSでメッセージが送信および受信された順序が確実に保持されるように、各プロデューサーはすべてのメッセージ送信で固有のメッセージグループIDを使用するようにしてください。

FIFOキューロジックはメッセージグループIDごとにのみ適用されます。各メッセージグループ ID は、 Amazon SQSキュー内の異なる順序付けされたメッセージグループを表します。メッセージグループIDごとに、すべてのメッセージが厳密な順序で送受信されます。ただし、異なるメッセージグループ ID値のメッセージは、送信および受信の順序が入れ替わる場合があります。メッセージグループIDをメッセージに関連付ける必要があります。メッセージグループIDを指定しない場合、アクションは失敗します。順序付けされたメッセージの単一グループが必要な場合は、FIFOキューに送信されるメッセージに同じメッセージグループ IDを指定します。

メッセージの受信

特定のメッセージグループIDのメッセージを受信するようにリクエストすることはできません。

FIFOキューから複数のメッセージグループIDを持つメッセージを受信した場合、Amazon SQSは最初に同じメッセージグループ IDを持つメッセージをできる限り多く返信するように試みます。こうすることで、他のコンシューマーが別のメッセージグループ IDを持つメッセージを処理できます。メッセージグループ ID があるメッセージを受信した場合、そのメッセージを削除しない、または表示されない限り、同じメッセージグループ ID のメッセージはそれ以上返されません。

> **Note**
>
> MaxNumberOfMessages アクションのReceiveMessageリクエストパラメータを使用した1回の呼び出しで最大10件のメッセージを受信できます。これらのメッセージには、FIFOの順序が保持され、同じメッセージグループ IDを持つことができます。したがって、同じメッセージグループ IDで使用可能なメッセージが10件未満の場合、別のメッセージグループ IDからメッセージを受信することがあります。10件のメッセージのバッチは同じですが、FIFOの順序は維持されます。

複数回の再試行

FIFOキューでは、プロデューサまたはコンシューマが複数の再試行を試みることができます:

- プロデューサが障害を検出したSendMessageアクションがある場合、同じメッセージ重複除外 IDを使用して、必要な回数だけ送信を再試行できます。重複除外間隔の期限が切れる前にプロデューサが少なくとも1つの確認応答を受信すると仮定すると、複数回の再試行はメッセージの順序に影響せず、重複も発生しません。
- コンシューマが障害を検出した場合ReceiveMessageアクションの場合、同じ受信リクエスト試行 IDを使用して、必要な回数だけ再試行できます。可視性タイムアウトの期限が切れる前にコンシューマが少なくとも1つの確認応答を受信すると仮定すると、複数回の再試行はメッセージの順序に影響しません。
- メッセージグループ IDがあるメッセージを受信した場合、メッセージを削除するか、表示されない限り、同じメッセージグループ IDのメッセージはそれ以上返信されません。

Amazon SQS での FIFO キューメッセージの順序付け

FIFOキューはスタンダードキューを改良したものであり、これを補完するものです。このキュータイプの最も重要な機能は、 FIFO (First-In-First-Out) FIFO 配信および1回だけの処理です。

- メッセージが送受信される順序は厳密に保持され、メッセージが 1 回配信され、コンシューマーが処理して削除するまで使用できなくなります。
- このキューでは、重複は導入されていません。

また、FIFOキューは、単一キュー内に複数の順序付けされたメッセージグループが存在できるメッセージグループをサポートします。FIFOキュー内のメッセージグループの数にクォータはありません。

Amazon SQS での 1 回限りの処理

スタンダードキューとは異なり、FIFOキューではメッセージの重複を招きません。FIFOキューを使用すると、重複をキューに送信することを防止するのに役立ちます。5分間の重複除外間隔内にSendMessageアクションを再試行しても、Amazon SQS ではキューに重複を招きません。

重複排除を設定するには、次のいずれかを実行する必要があります。

- コンテンツベースの重複排除を有効にします。これは、Amazon SQSにSHA-256ハッシュを使用して、メッセージ本文を使用したメッセージ重複除外ID(ただしメッセージの属性ではない) を生成するように指示するものです。詳細については、「CreateQueue、GetQueueAttributes、 およびSetQueueAttributes内のアクションAmazon Simple Queue Service APIリファレンスのドキュメント」を参照してください。
- メッセージに明示的にメッセージ重複排除 IDを指定 (またはシーケンス番号を参照) します。詳細については、「SendMessage、SendMessageBatch、 およびReceiveMessage内のアクションAmazon Simple Queue Service APIリファレンスのドキュメント」を参照してください。

Amazon SQS の標準キューからFIFOキューへの移行

スタンダード キューを使用している既存のアプリケーションがあり、FIFO キューの特徴である順序付けや 1 回のみ処理を活用する場合、キューとアプリケーションを正しく設定する必要があります。

> **Note**
>
> 既存のスタンダードキューをFIFOキューに変換することはできません。この移行を行うには、アプリケーション用に新しい FIFO キューを作成するか、既存のスタンダードキューを削除して FIFO キューとして再作成する必要があります。

アプリケーションがFIFOキューを使用して正しく動作することを確認するために次のチェックリストを使用すること。

- スループットを向上させるには、FIFO に推奨される高スループットモードを使用してください。メッセージングのクォータの詳細については、「Amazon SQS メッセージクォータ」を参照してください。

- FIFOキューでは、メッセージ単位の遅延をサポートしていません。キュー単位の遅延のみをサポートします。アプリケーションで、各メッセージのDelaySecondsパラメータが同じ値に設定されている場合、アプリケーションを変更してメッセージ単位の遅延を削除し、代わりにキュー全体にDelaySecondsを設定する必要があります。
- メッセージグループは、お客様がそれぞれの順序を維持しながらメッセージを並列処理できるようにする独自の FIFO 機能です。お客様はメッセージグループ ID を指定してメッセージをメッセージグループに整理します。メッセージグループは、多くの場合、特定のワークロードのビジネスディメンションに基づいています。FIFO キューでの拡張性を高めるには、メッセージ ID のビジネスディメンションをより細かく設定してください。メッセージを配信するメッセージグループ ID が多いほど、FIFO が使用できるメッセージの数も増えます。
- FIFOキューにメッセージを送信する前に、以下を確認してください。
  - アプリケーションが同一のメッセージ本文を使用してメッセージを送信する場合は、アプリケーションを変更して、送信される各メッセージに対して一意のメッセージ重複排除 IDを提供できます。
  - アプリケーションが一意のメッセージ本文を使用してメッセージを送信する場合は、コンテンツベースの重複排除を有効にできます。
- コンシューマーのコードを変更する必要はありません。ただし、メッセージの処理に長時間かかり、可視性タイムアウトが高い値に設定されている場合は、各ReceiveMessage アクションに受信リクエスト試行 ID を追加することを検討してください。これは、ネットワーク障害の場合に受信試行を再試行し、受信試行の失敗によりキューが停止するのを防止します。

詳細については、 Amazon Simple Queue Service APIリファレンスを参照してください。

Amazon SQS の FIFO キューの高スループット

Amazon SQS高スループットFIFOキューは、厳密なメッセージ順序を維持しながら高メッセージスループットを効率的に管理し、多数のメッセージを処理するアプリケーションの信頼性とスケーラビリティを確保します。このソリューションは、高スループットと順序付けられたメッセージ配信の両方を必要とするシナリオに最適です。

Amazon SQS 高スループット FIFO キューは、厳密なメッセージの順序付けが重要ではなく、受信メッセージの量が比較的少ない、または散発的なシナリオでは必要ありません。例えば、低頻度または非連続のメッセージを処理する小規模なアプリケーションがある場合、高スループットの FIFO キューに関連する複雑さとコストの増加は正当化されない可能性があります。さらに、アプリケーションで高スループット FIFO キューが提供する拡張スループット機能が必要ない場合は、標準の Amazon SQS キューを選択することで、コスト効率が高く、管理が簡単になる可能性があります。

高スループットの FIFO キューでリクエストキャパシティーを強化するには、メッセージグループの数を増やすことをお勧めします。高スループットのメッセージクォータの詳細については、「Amazon Web Services 全般のリファレンス」の「Amazon SQS Service Quotas」を参照してください。

キューごとのクォータとデータ分散戦略については、Amazon SQS メッセージクォータ「」および「」を参照してくださいSQS FIFOキューの高スループットを実現するパーティションとデータ分散。

トピック

- Amazon SQS FIFO キューの高スループットのユースケース
- SQS FIFOキューの高スループットを実現するパーティションとデータ分散
- Amazon SQS で FIFO キューの高スループットを有効にする

Amazon SQS FIFO キューの高スループットのユースケース

次のユースケースでは、高スループット FIFO キューのさまざまなアプリケーションに焦点を当て、業界やシナリオ全体での有効性を示します。

1. リアルタイムデータ処理: イベント処理やテレメトリデータインジェストなど、リアルタイムデータストリームを処理するアプリケーションは、スループットの高い FIFO キューを利用して、メッセージの継続的な流入を処理しながら、正確な分析のために順序を維持することができます。
2. e コマース注文処理： 顧客取引の順序を維持することが重要な e コマースプラットフォームでは、高スループットの FIFO キューにより、ショッピングのピーク時でも注文が遅延なく順番に処理されます。
3. 金融サービス： 高頻度の取引データまたはトランザクションデータを処理する金融機関は、高スループットの FIFO キューに依存して、メッセージの順序付けに関する厳格な規制要件に準拠しながら、レイテンシーを最小限に抑えながら市場データとトランザクションを処理します。
4. メディアストリーミング: ストリーミングプラットフォームとメディアディストリビューションサービスは、高スループットの FIFO キューを使用してメディアファイルとストリーミングコンテンツの配信を管理します。これにより、コンテンツ配信の正しい順序を維持しながら、ユーザーのスムーズな再生エクスペリエンスを確保できます。

SQS FIFOキューの高スループットを実現するパーティションとデータ分散

Amazon SQSは、FIFOキューデータをパーティションに保存します。パーティションは、 AWS リージョン内の複数のアベイラビリティーゾーンに自動的にレプリケートされるキューのストレージの割り当てです。パーティションは管理しません。代わりに Amazon SQSがパーティション管理を処理します。

FIFOキューの場合、Amazon SQSは次の状況でキューのパーティションの数を変更します。

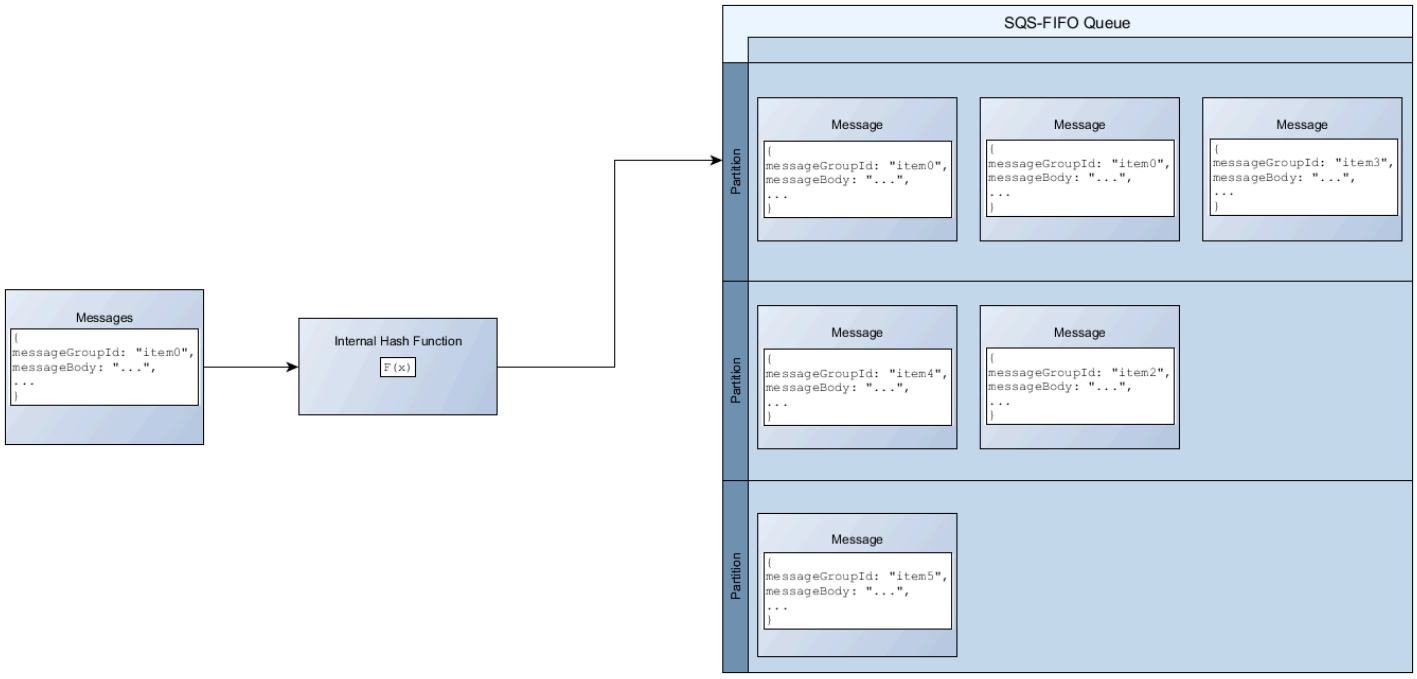
- 現在のリクエストレートが既存のパーティションがサポートできる値に近づいたり超えたりすると、キューがリージョナルクォータに達するまで追加のパーティションが割り当てられます。クオータの詳細については、{Amazon SQS メッセージクォータ}を参照してください。
- 現在のパーティションの使用率が低い場合は、パーティションの数が減ることがあります。

パーティション管理は自動的にバックグラウンドで自動的に発生し、アプリケーションに対して透過的です。キューとメッセージは常に利用可能です。

メッセージグループ ID によるデータの配布

FIFOキューにメッセージを追加するには、Amazon SQSは各メッセージのメッセージグループ ID の値を内部ハッシュ関数への入力として使用します。ハッシュ関数からの出力値によって、どのパーティションにメッセージが保存されるが決まります。

次の図は、複数のパーティションにまたがるキューを示しています。キューのメッセージグループIDは、アイテム番号に基づきます。Amazon SQSは、ハッシュ関数を使用して、新しい項目の保存場所を決定します。この場合は、文字列のハッシュ値に基づいていますitem0。アイテムは、キューに追加される順序と同じ順序で格納されることに注意してください。各アイテムの場所は、メッセージグループIDのハッシュ値によって決まります。



> **Note**
>
> Amazon SQS は、パーティションの数に関係なく、FIFO キューのパーティション間で項目を均一に分散 AWS するように最適化されています。多数の異なる値を持つことができるメッセージグループ IDs を使用することをお勧めします。

パーティション使用率の最適化

サポートされているリージョンにおいては、各パーティションは、バッチ処理により 1 秒あたり最大 3,000 件のメッセージ、または送信、受信、削除操作では 1 秒あたり最大 300 件のメッセージをサポートします。高スループットのメッセージクォータの詳細については、「Amazon Web Services 全般のリファレンス」の「Amazon SQS Service Quotas」を参照してください。

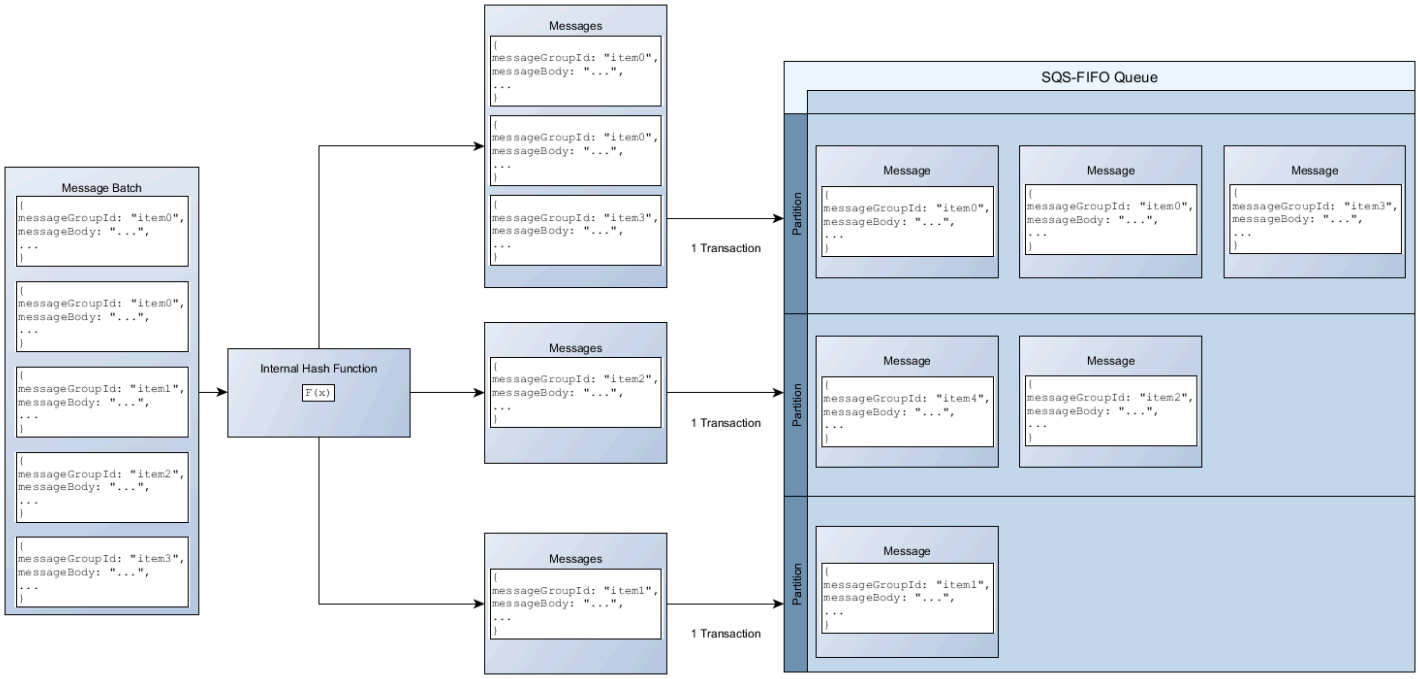
バッチAPI を使用する場合、各メッセージはメッセージグループ ID によるデータの配布で説明されているプロセスに基づいてルーティングされます。同じパーティションにルーティングされたメッセージは、単一のトランザクションでグループ化され、処理されます。

`SendMessageBatch` API のパーティション使用率を最適化するために、 AWS では、可能な場合は同じメッセージグループ IDs でメッセージをバッチ処理することをお勧めします。

`DeleteMessageBatch` および `ChangeMessageVisibilityBatch` APIs、 AWS は `MaxNumberOfMessages`パラメータを 10 に設定して`ReceiveMessage`リクエストを使用し、1 つ

のReceiveMessageリクエストによって返される受信ハンドルをバッチ処理することをお勧めします。

次の例では、さまざまなメッセージグループ IDを持つメッセージのバッチが送信されます。バッチは3つのグループに分割され、それぞれがパーティションのクォータに対してカウントされます。



> **Note**
>
> Amazon SQS では、同じメッセージグループ ID の内部ハッシュ関数を持つメッセージがバッチリクエスト内でグループ化されている場合にのみ保証されます。内部ハッシュ関数の出力とパーティションの数に応じて、異なるメッセージグループ IDを持つメッセージがグループ化されることがあります。ハッシュ関数またはパーティションの数はいつでも変更できるため、ある時点でグループ化されたメッセージは後でグループ化されない場合があります。

Amazon SQS で FIFO キューの高スループットを有効にする

新規または既存の FIFO キューに対して高スループットを有効にできます。この機能には、FIFO キューを作成および編集するときに、次の 3つの新しいオプションが追加されます。

- 高スループット FIFO を有効にする—現在の FIFOキューのメッセージに対し、より高いスループットを有効にします。

- 重複除外スコープ-重複除外をキューレベルまたはメッセージグループレベルのどちらで実行するかを指定します。
- FIFO スループット制限—FIFO キューのメッセージのスループットクォータをキューまたはメッセージグループレベルのどちらで設定するかを指定します。

FIFO キュー(コンソール)の高スループットを有効にするには

1. 作成を起動またはFIFO キューを編集。
2. キューのオプションを指定するときは、高スループット FIFO を有効にする。

   FIFOキューの高スループットを有効にすると、関連するオプションは次のように設定されます。

   - 重複除外スコープは、FIFOキューの高スループットを使用するために必要な設定のメッセージグループ に設定されます。
   - FIFO スループット制限は、FIFO キューの高スループットを使用するために必要な設定のメッセージグループ ID単位 に設定されます。

   FIFO キューの高スループットを使用するために必要な設定のいずれかを変更すると、通常のスループットがキューに対して有効になり、指定どおりに重複除外が実行されます。
3. キューのすべてのオプションをの指定を継続 終了したら、キューの作成または保存を選択します。

FIFOキューを作成または編集した後、メッセージの送信およびメッセージの受信と削除、全てをより高いTPSで行うことができます。高スループットクォータについては、「Amazon SQS メッセージクォータ」の「メッセージスループット」を参照してください。

Amazon SQS の主要な用語

以下の重要な用語は、FIFOキューの機能をよりよく理解するうえで役立ちます。詳細については、「Amazon Simple キューService APIリファレンス」を参照してください。

メッセージ重複排除ID

送信されたメッセージの重複除外に使用されるトークン。特定のメッセージ重複除外ID を持つメッセージが正常に送信されると、同じメッセージ重複除外ID で送信されたメッセージは正常に受け入れられますが、5分間の重複除外間には配信されません。

> **Note**
>
> Amazon SQSは、メッセージを受信して削除した後も、メッセージの重複除外IDを追跡し続けます。

メッセージグループ ID

タグ 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特定する必須トークンです。同じメッセージグループに属するメッセージは、常にメッセージグループに対して厳密な順序で1つずつ処理されます (ただし、異なるメッセージグループに属するメッセージは、順不同に処理されます) 。

受信リクエスト試行ID

トークンは重複除外のReceiveMessage 呼び出しに使用されます。

シーケンス番号

Amazon SQS が各メッセージに割り当てる、連続しないラージ番号。

Amazon SQS での FIFO 互換性

クライアント

Amazon SQSバッファリング非同期クライアントは現在 FIFOキューをサポートしていません。

サービス

アプリケーションが複数の AWS サービス、または AWS と外部サービスを組み合わせて使用する場合は、どのサービス機能が FIFO キューをサポートしていないかを理解することが重要です。

Amazon SQS に通知を送信する一部の AWS または外部サービスは、FIFO キューをターゲットとして設定できるにもかかわらず、FIFO キューと互換性がない場合があります。

現在、 AWS サービスの以下の機能は FIFO キューと互換性がありません。

- Amazon S3イベント通知
- Auto Scalingライフサイクルフック
- AWS IoT ルールアクション

- AWS Lambda デッドレターキュー

FIFOキューと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 SQS の FIFO キューとメッセージ識別子

このセクションでは、FIFO キューの識別子について説明します。これらの識別子は、特定のキューとメッセージを見つけて操作するうえで役立ちます。

トピック

- Amazon SQSのFIFOキューの識別子
- Amazon SQS FIFOキューのその他の識別子

Amazon SQSのFIFOキューの識別子

次の識別子の詳細については、Amazon Simpleキューサービス APIリファレンス。を参照してください

キュー名およびURL

新しいキューを作成する際は、 AWS アカウントおよびリージョンに一意なキュー名を指定する必要があります。Amazon SQSは、作成した各キューをキュー URLこれには、キュー名と他のAmazon SQSコンポーネントが含まれます。キューでアクションを実行するときは必ず、そのキュー URLを指定します。

FIFOキューの名前は.fifoのサフィックスで終わる必要があります。サフィックスは80文字のキュー名クォータにカウントされます。キューがであるかどうかを確認するにはFIFOでは、キュー名の末尾がサフィックスで終わるかどうかでチェックすることがきます。

以下は、AWS アカウント番号 を持つユーザーがMyQueue所有する という名前の FIFO キューのキュー URL です123456789012。

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue.fifo
```

キューを一覧表示し、アカウント番号に続く文字列を解析することで、キューのURLをプログラムで取得できます。詳細については、「ListQueues」を参照してください。

メッセージ ID

各メッセージは、システムにより割り当てられたメッセージ ID を受け取り、Amazon SQS は`SendMessage` レスポンスにより返信します。この識別子は、メッセージを特定する場合に役立ちます。メッセージ IDの最大長は100文字です。

受信ハンドル

キューからメッセージを受信するたびに、そのメッセージの受信ハンドルを受け取ります。このハンドルは、メッセージ自体ではなくメッセージ受信のアクションと関連付けられます。メッセージを削除したり、メッセージ可視性を変更したりするには、受信ハンドル (メッセージ ID ではなく) を指定する必要があります。つまり、メッセージを削除する前にメッセージを受信する必要があります (メッセージをキューにおいてから回収することはできません)。受信ハンドルの最大長は1,024文字です。

> **Important**
>
> メッセージを複数回受信した場合、受信するたびに異なる受信ハンドルを受け取ります。メッセージの削除をリクエストするときは、最後に受け取った受信ハンドルを指定してください (そうしないと、メッセージが削除されない可能性があります)。

受信ハンドルの例を次に示します (3行に分割されています)。

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

Amazon SQS FIFOキューのその他の識別子

次の識別子の詳細については、Amazon SQS での 1 回限りの処理とAmazon Simple キュー サービス APIリファレンスを参照してください

メッセージ重複排除ID

送信されたメッセージの重複除外に使用されるトークン。特定のメッセージ重複除外ID を持つメッセージが正常に送信されると、同じメッセージ重複除外ID で送信されたメッセージは正常に受け入れられますが、5分間の重複除外間には配信されません。

メッセージグループ ID

タグ 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特定する必須トークンです。同じメッセージグループに属するメッセージは、常にメッセージグループに対して厳密な順序で1通ずつ処理されます (ただし、異なるメッセージグループに属するメッセージは、順不同に処理されます)。

シーケンス番号

Amazon SQS が各メッセージに割り当てる、連続しないラージ番号。

Amazon SQSクォータ

このトピックでは、Amazon Simple Queue Service (Amazon SQS) 内のクォータを一覧表示します。

トピック

Amazon SQS FIFO キュークォータ

Amazon SQSクォータ

次の表に、FIFO キューに関連するクォータの一覧を示します。

クォータ	説明
遅延キュー	キューのデフォルト (最小) 遅延は0秒です。最大は15分です。
表示されたキュー	`ListQueues` リクエストあたり1,000キュー。
ロングポーリングの待ち時間	ロングポーリングの最大待機時間は20秒です。
メッセージグループ	FIFOキュー内のメッセージグループの数にクォータはありません。
キューあたりのメッセージ (バックログ)	Amazon SQSキューに保存できるメッセージ数は無制限です。
キューあたりのメッセージ (処理中)	FIFO キューでは、最大 20,000 通 の処理中メッセージ (コンシューマがキューから受信して、キューから削除されていないもの) が存在する可能性があります。このクォータに届いた場合、Amazon SQSはエラーメッセージを返しません。

クォータ	説明
キュー名	FIFOキューの名前は.fifoのサフィックスで終わる必要があります。サフィックスは80文字のキュー名クォータにカウントされます。キューがあるかどうかを確認するにはFIFOでは、キュー名の末尾がサフィックスで終わるかどうかをチェックできます。
キューのタグ	50個を超えるタグをキューに追加することはお勧めしません。タグは、UTF-8 の Unicode 文字をサポートします。
	タグKeyは必須ですが、タグValueはオプションです。
	タグKeyとタグValueの大文字と小文字は区別されます。
	タグKeyとタグValueには、UTF-8のUnicode 英数字と空白を含めることができます。次の特殊文字を使用できます。_ . : / = + - @
	タグKeyまたはタグValueには、予約済みプレフィックスのaws:を含めることはできません (このプレフィックスが付いているタグキーやタグ値は削除できません)。
	タグKeyの最大長はUTF-8で128Unicode文字です。タグKeyを空またはnullにすることはできません。
	タグValueの最大長はUTF-8で256Unicode文字です。タグValueを空またはnullにすることはできません。
	タグ付けアクションは、 あたり 30 TPS に制限されています AWS アカウント。アプリケーションでより高いスループットが必要な場合、リクエストを送信します。。

Amazon SQS メッセージクォータ

次の表に、メッセージに関連するクォータの一覧を示します。

クォータ	説明
バッチ処理されたメッセージ ID	バッチ処理されたメッセージ ID には最大80文字を使用できます。次の文字を使用できます。英数字、ハイフン (-)、およびアンダースコア (_)。
メッセージ属性	メッセージには、最大10個のメタデータ属性を含めることができます。
メッセージバッチ	ひとつのメッセージBatchリクエストに最大10個のメッセージを含めることができます。詳細については、AmazonSQSBufferedAsync クライアントの設定 セクションの Amazon SQSのバッチアクション を参照してください。
メッセージの内容	メッセージには、XML、JSON、およびフォーマットされていないテキストのみを含めることができます。次の Unicode 文字を使用できます。#x9 | #xA | #xD | #x20から#xD7FF | #xE000から#xFFFD | #x10000から#x10FFFF この一覧に含まれていない文字は、すべて拒否されます。詳細については、文字に関するW3C仕様を参照してください。
メッセージグループ ID	同じメッセージグループ IDを持つメッセージのラージバックログを構築しないように、バックログからメッセージを消費します。 MessageGroupId はFIFOキューには必須です。これをスタンダードキューには使用できません。 空でないMessageGroupId メッセージ付きを関連づける必要があります。MessageGroupId を指定しない場合、アクションは失敗します。

クォータ	説明
	MessageGroupId の最大長は128文字です。有効な値: 英数字と句読点(!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~) 。
メッセージの保持	デフォルトでは、メッセージは4日間保持されます。最小の期間は60秒(1分) です。最大は1,209,600秒(14日)です。
メッセージのスループット	スタンダードキューは、API アクション (SendMessage 、ReceiveMessage 、または DeleteMessage) ごとに1秒あたりほぼ無制限の API コールをサポートします。
	FIFOキュー • FIFO キューは、API アクション (SendMessage 、ReceiveMessage 、または DeleteMessage) ごとに 1 秒あたり 300 トランザクションの割り当てをサポートします。 • バッチ処理を使用する場合、FIFO キューは API メソッド (SendMessage 、ReceiveMessage 、または DeleteMessage) ごとに 1 秒あたり最大 3,000 通のメッセージをサポートします。1 秒あたり 3,000 通のメッセージは 300 回の API コールを表し、それぞれに 10 個のメッセージのバッチがあります。

クォータ	説明
	FIFOキューの高スループット • バッチ処理 (SendMessage 、ReceiveMessage 、または DeleteMessage) を使用しない場合、FIFO キューの高スループットでは、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド) の各リージョンで、API アクションごとに、1 秒あたり最大 70,000 トランザクションが処理されます。 • 米国東部 (オハイオ) リージョンと欧州 (フランクフルト) リージョンでは、デフォルトのスループットは API アクションごとに 1 秒あたり 18,000 トランザクションです。 • アジアパシフィック (ムンバイ)、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) の各リージョンでは、デフォルトのスループットは、API アクションごとに、1 秒あたり 9,000 トランザクションです。 • 欧州 (ロンドン) と南米 (サンパウロ) では、デフォルトのスループットは API アクションごとに 1 秒あたり 4,500 トランザクションです。 • 最大スループットを得るためには、バッチ処理なしで送信されるメッセージに使用するメッセージグループ IDの数を増やします。 • 米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド) の各リージョンでバッチ処理 API (SendMessageBatch または DeleteMessageBatch) を使用すると、スループットを 1 秒あたり 700,000 メッセージまで増やすことができます。1 秒あたり 700,000 通のメッセージ数は 1 秒あたり 70,000 回のトランザクションを表し、それぞれに 10 個のメッセージのバッチがあります。 欧州 (フランクフルト) リージョンと米国東部 (オハイオ) リージョンでは、バッチ処理 API を使用すれば 1

クォータ	説明
	秒あたり最大 180,000 件のメッセージを処理できます。1 秒あたり 180,000 通のメッセージ数は 1 秒あたり 18,000 回のトランザクションを表し、それぞれに 10 個のメッセージのバッチがあります。 アジアパシフィック (ムンバイ)、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) の各リージョンでは、バッチ処理では、1 秒あたり 90,000 メッセージまで処理できます。使用時に最大スループットを達成するには`SendMessageBatch` そして`DeleteMessageBatch` では、batchリクエスト内のすべてのメッセージに、同じメッセージグループ ID を使用する必要があります。 • 欧州 (ロンドン) と南米 (サンパウロ) の各リージョンでは、バッチ処理により 1 秒あたり最大 45,000 件のメッセージを処理できます。使用時に最大スループットを達成するには`SendMessageBatch` そして`DeleteMessageBatch` では、batchリクエスト内のすべてのメッセージに、同じメッセージグループ ID を使用する必要があります。 • 他のすべての AWS リージョンでは、API アクションごとに、最大スループットは 2,400 (バッチ処理なし) または 24,000 (バッチ処理を使用) メッセージ/秒です。 • リージョン制限を超えるクォータの引き上げをリクエストするには、サポートリクエスト を送信します。 • 詳細については、「SQS FIFOキューの高スループットを実現するパーティションとデータ分散」を参照してください。
メッセージタイマー	メッセージのデフォルト (最小) 遅延は0秒です。最大は15分です。

クォータ	説明
メッセージサイズ	最小メッセージサイズは1バイト(1文字) です。最大は262,144 バイト (256 KiB) です。 256 KiB を超えるメッセージを送信するには、Java 用 Amazon SQS 拡張クライアントライブラリと Python 用 Amazon SQS 拡張クライアントライブラリを使用できます。このライブラリでは、 Amazon SQSのメッセージペイロードAmazon S3へのリファレンスを含むメッセージを送信できます。最大ペイロードサイズは2GBです。 **Note** この拡張ライブラリは同期クライアントでのみ機能します。
メッセージ可視性タイムアウト	デフォルトの可視性タイムアウトは30秒です。最小は0秒です。最大は12時間です。
ポリシー情報	最大クォータは8,192バイト、20ステートメント、50プリンシパル、または 10条件になります。詳細については、「Amazon SQS ポリシークォータ」を参照してください。

Amazon SQS ポリシークォータ

次の表に、ポリシーに関連するクォータの一覧を示します。

名前	最大値
バイト	8,192
条件	10
プリンシパル	50

名前	最大値
ステートメント	20
ステートメントあたりのアクション	7

Amazon SQS の特徴と機能

Amazon SQS には、以下の特徴と機能があります。

トピック

Amazon SQS でデッドレターキューを使用します

Amazon SQS はデッドレターキュー (DLQsをサポートしています。このキューは、正常に処理されないメッセージを対象にできます。DLQsは、未使用のメッセージを分離して処理が成功しなかった理由を判断できるため、アプリケーションのデバッグに役立ちます。最適なパフォーマンスを得るには、ソースキューと DLQ を同じ AWS アカウント とリージョン内に保持するのがベストプラクティスです。メッセージがデッドレターキューに入ると、次のことが可能になります。

- デッドレターキューにメッセージが移動する原因となった例外を口グを調べる。
- デッドレターキューに移動したメッセージの内容を分析して、アプリケーションの問題を診断します。
- コンシューマーがメッセージを処理するために十分な時間が設定されているかどうかを調べる。
- デッドレターキューリドライブ を使用して、デッドレターキューからメッセージを移動します。

デッドレターキューとして設定する前に、まず新しいキューを作成する必要があります。Amazon SQS コンソールを使用したデッドレターキューの設定についての詳細は、「Amazon SQS コンソールを使用してデッドレターキューを設定する方法について説明します。」を参照してください。デッドレターキューに移動されたメッセージのアラームを設定する方法など、デッドレターキューに関するヘルプについては、「」を参照してくださいAmazon を使用してデッドレターキューのアラームを作成する CloudWatch。

デッドレターキューのポリシーの使用

リドライブポリシーを使用して を指定しますmaxReceiveCount。maxReceiveCount は、コンシューマーがデッドレターキューに移動する前にソースキューからメッセージを受信できる回数です。例えば、 maxReceiveCountが 1 などの低い値に設定されている場合、メッセージの受信に 1 回失敗すると、メッセージがデッドレターキューに移動します。システムがエラーに対して回復力を持つようにするには、maxReceiveCount十分な再試行ができるように高めに設定してください。

リドライブ許可ポリシーはデッドレターキューにアクセスできるソースキューを指定します。すべてのソースキューを許可するか、特定のソースキューを許可するか、すべてのソースキューによるデッドレターキューの使用を拒否するかを選択できます。デフォルトでは、すべてのソースキューがデッドレターキューを使用できます。byQueue オプションを使用して特定のキューを許可する場合は、ソースキューの Amazon リソースネーム (ARN) を使用して最大 10 個のソースキューを指定できます。denyAllを指定した場合、キューをデッドレターキューとして使用することはできません。

デッドレターキューのメッセージ保持期間を理解する

スタンダードキューでは、メッセージの有効期限は、常に元のエンキューのタイムスタンプに基づきます。デッドレターキューに移動すると、エンキューのタイムスタンプは変更されません。ApproximateAgeOfOldestMessage メトリクスは、メッセージが最初に送信された時刻ではなく、メッセージがデッドレターキューに移動された時刻を示します。たとえば、メッセージがデッドレターキューに移動される前に、元のキューで1日費やすと仮定します。デッドレターキューの保持期間が4日間である場合、メッセージは3日後にデッドレターキューから削除され、ApproximateAgeOfOldestMessageは3日間です。したがって、デッドレターキューの保持期間を、元のキューの保持期間よりも長く設定することがベストプラクティスです。

FIFO キューでは、メッセージがデッドレターキューに移動すると、エンキューのタイムスタンプがリセットされます。ApproximateAgeOfOldestMessage メトリクスは、メッセージがデッドレターキューに移動した日を示します。上記の同じ例では、メッセージは 4 日後にデッドレターキューから削除され、ApproximateAgeOfOldestMessage は 4 日間です。

Amazon SQS コンソールを使用してデッドレターキューを設定する方法について説明します。

デッドレターキューは、ソースキューが正常に処理されないメッセージをターゲットにできるキューです。詳細については、「Amazon SQS でデッドレターキューを使用します 」を参照してください。

Amazon SQSはデッドレターキューを自動的に作成しません。デッドレターキューとして使用する前に、最初にキューを作成する必要があります。デッドレターキューとして使用するキューを作成する手順については、「」を参照してくださいAmazon SQS コンソールを使用してキューを作成する。

FIFOキューのデッドレターキューは、FIFOキューでもある必要があります。同様に、標準キューのデッドレターキューは、標準キューでもある必要があります。

キューを作成または編集する場合、デッドレターキューを設定できます。

既存のキュー(コンソール)のデッドレターキューを設定する

1. Amazon SQSコンソールを開きますhttps://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)]を選択します。
3. キューを選択し、編集 を選択します。
4. {デッドレターキュー}セクションをまでスクロールし、[Enabled (有効)]を選択します。
5. このソースキューに関連付ける既存のデッドレターキューの Amazon リソースネーム (ARN) を選択します。
6. デッドレターキューに送信されるまでにメッセージを受信できる最大回数を設定するには、{Maximum Receives}を 1から1,000 の値に設定します。
7. デッドレターキューの設定が完了したら、保存を選択します。

   キューを保存すると、コンソールはキューの詳細ページを表示します。詳細ページでデッドレターキュータブは、最大受信数そしてデッドレターキューのARNデッドレターキューを表示します。

Amazon SQS でデッドレターキューリドライブを設定する方法について説明します。

デッドレターキューリドライブを使用して、未使用のメッセージを既存のデッドレターキューから移動できます。デフォルトでは、デッドレターキューのリドライブは、デッドレターキューからソースキューにメッセージを移動します。ただし、どちらのキューも同じタイプであれば、他のキューをリドライブの宛先として設定することもできます。例えば、デッドレターキューが FIFO キューの場合、リドライブの宛先キューも FIFO キューでなければなりません。さらに、リドライブベロシティを設定してAmazon SQSがメッセージを移動するレートを設定できます。

> **Note**
>
> メッセージが FIFO キューから FIFO DLQ に移動すると、元のメッセージの重複排除 ID は元のメッセージの ID に置き換えられます。これは、重複排除 ID を共有している 2 つの独立したメッセージが DLQ 重複排除によって保存されなくなることがないようにするためです。

デッドレターキューは、最も古いメッセージから順にメッセージをリドライブします。ただし、宛先キューは、リドライブされたメッセージと他のプロデューサーからの新しいメッセージを、受信した順序に従って取り込みます。例えば、プロデューサーがデッドレターキューからリドライブされたメッセージを同時に受信するときにソース FIFO キューにメッセージを送信している場合、リドライブされたメッセージはプロデューサーからの新しいメッセージと交差します。

> **Note**
>
> リドライブタスクによって、保持期間がリセットされます。すべてのリドライブされたメッセージは、新しい を持つ新しいメッセージと見なmessageIDされenqueueTime、リドライブされたメッセージに割り当てられます。

トピック

- Amazon SQS API を使用した既存の標準キューのデッドレターキューリドライブの設定
- Amazon SQS コンソールを使用した既存の標準キューのデッドレターキューリドライブの設定
- デッドレターキューリドライブのキューアクセス許可を設定

Amazon SQS API を使用した既存の標準キューのデッドレターキューリドライブの設定

デッドレターキューのリドライブはSendMessageBatch、、ReceiveMessage、およびDeleteMessageBatch API アクションを使用して設定できます。

API アクション	説明
StartMessageMoveTask	指定されたソースキューから指定された送信先キューにメッセージを移動する非同期タスクを開始します。
ListMessageMoveTasks	特定のソースキューにある最新のメッセージ移動タスク (最大 10 個) を取得します。
CancelMessageMoveTask	指定されたメッセージ移動タスクをキャンセルします。メッセージの移動は、現在のステータスが RUNNING の場合にのみキャンセルできます。

Amazon SQS コンソールを使用した既存の標準キューのデッドレターキューリドライブの設定

1. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)] を選択します。
3. デッドレターキューとして設定されたキューの名前を選択します。
4. DLQリドライブを開始するを選択する。
5. リドライブの設定に対してメッセージの送信先で、次のいずれかを実行します。
   - メッセージをソースキューに戻すには、ソースキューにリドライブを選択します。
   - メッセージを別のキューにリドライブするには、カスタム送信先にリドライブを選択します。次に、既存の宛先キューのAmazonリソースネーム (ARNを入力します。
6. Velocityコントロール設定で、次のいずれかを選択します。
   - システム最適化-デッドレターキューメッセージを1秒につき最大メッセージ数でリドライブします。

- カスタム最大 velocity-1秒につきカスタム最大メッセージ数でデッドレターキューメッセージをリドライブします。許可される最大レートは1秒に500 メッセージです。
  - カスタム最大速度を小さい値から始めて、ソースキューがメッセージでいっぱいにならないことを確認することをお勧めします。そこから、ソースキューの状態を引き続き監視しながら、カスタム最大速度の値を徐々に上げていきます。

7. デッドレターキューリドライブの設定が完了したら、メッセージをリドライブするを選択します。

> **Important**
>
> Amazon SQSは、デッドレターキューからメッセージをリドライブするときにメッセージのフィルタリングと変更をサポートしていません。
> デッドレターキューのリドライブタスクは、最大36時間実行できます。Amazon SQSでは、アカウントごとに最大100通のアクティブなリドライブタスクがサポートされます。

8. メッセージリドライブタスクをキャンセルする場合は、の詳細キューのページで、DLQリドライブをキャンセルします。進行中のメッセージのリドライブをキャンセルすると、移動先キューにすでに正常に移動されたメッセージは、移動先キューに残ります。

デッドレターキューリドライブのキューアクセス許可を設定

ポリシーにアクセス許可を追加することで、ユーザーに特定のデッドレターキューアクションへのアクセスを許可できます。デッドレターキューリドライブに最低限必要なアクセス許可は次のとおりです。

最小限必要なアクセス権限	必要な API メソッド
メッセージのリドライブを開始するには	• デッドレターキューの `sqs:StartMessageMoveTask` 、`sqs:ReceiveMessage` 、`sqs:DeleteMessage` 、`sqs:GetQueueAttributes` を追加します。デッドレターキューまたは元のソースキュー (SSE キューとも呼ばれます) のいずれかが暗号化されている場合、`kms:Decrypt` メッセージの暗号化に使用されたすべての KMS キーも必要になります。

最小限必要なアクセス権限	必要な API メソッド
	• 送信先キュー `sqs:SendMessage` を追加します。送信先キューが暗号化されている場合、`kms:GenerateDataKey` と `kms:Decrypt` も必要になります。
進行中のメッセージのリドライブをキャンセルするには	• デッドレターキューの `sqs:CancelMessageMoveTask` 、`sqs:ReceiveMessage` 、`sqs:DeleteMessage` 、`sqs:GetQueueAttributes` を追加します。デッドレターキューが暗号化されている場合 (SSE キューとも呼ばれる)、`kms:Decrypt` も必要になります。
メッセージの移動状況を表示するには	• デッドレターキューの `sqs:ListMessageMoveTasks` と `sqs:GetQueueAttributes` を追加します。

暗号化されたキューペア (デッドレターキューのあるソースキュー) のアクセス許可を設定するには

次の手順を使用して、デッドレターキューリドライブの最小限のアクセス許可を設定します。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインで、ポリシー を選択します。
3. 次のアクセス許可を持つポリシーを作成して、ログイン IAM ユーザーまたはロールにアタッチします。

   - `sqs:StartMessageMoveTask`
   - `sqs:CancelMessageMoveTask`
   - `sqs:ListMessageMoveTasks`
   - `sqs:ListDeadLetterSourceQueues`
   - `sqs:ReceiveMessage`
   - `sqs:DeleteMessage`
   - `sqs:GetQueueAttributes`
   - デッドレターキューの `Resource` ARN (例えば、"arn:aws:sqs:*<DLQ_region>*:*<DLQ_accountId>*:*<DLQ_name>*")。

- sqs:SendMessage
- 送信先キューの Resource ARN (例：「arn:aws:sqs:*<DestQueue_region>:<DestQueue_accountId >:<DestQueue_name>*」)。
- kms:Decrypt — 復号アクションを許可します。
- kms:GenerateDataKey
- 元のソースキュー内のメッセージの暗号化に使用された任意の KMS 暗号化キーの Resource ARN (例えば、"arn:aws:kms:*<region>*:*<accountId>*:key/*<keyId_used to encrypt the message body>*")。
- リドライブ送信先キューに使用される KMS 暗号化キーのリソース ARN (例えば、"arn:aws:kms:*<region>*:*<accountId>*:key/*<keyId_used for the destination queue>*")。

アクセスポリシーは以下のようになります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:StartMessageMoveTask",
        "sqs:CancelMessageMoveTask",
        "sqs:ListMessageMoveTasks",
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage",
        "sqs:GetQueueAttributes",
        "sqs:ListDeadLetterSourceQueues"
      ],
      "Resource": "arn:aws:sqs:<DLQ_region>:<DLQ_accountId>:<DLQ_name>"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource":
 "arn:aws:sqs:<DestQueue_region>:<DestQueue_accountId>:<DestQueue_name>"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "arn:aws:kms:<region>:<accountId>:key/<keyId>"
    }
  ]
}
```

暗号化されていないキューペア (デッドレターキューのあるソースキュー) を使用してアクセス許可を設定するには

次の手順を使用して、暗号化されていない標準的なデッドレターキューの最小限のアクセス許可を設定します。最低限必要なアクセス許可は、デッドレターキューからの属性の受信、削除、取得、およびソースキューへの属性の送信です。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ナビゲーションペインで、ポリシー を選択します。
3. 次のアクセス許可を持つポリシーを作成して、ログイン IAM ユーザーまたはロールにアタッチします。

   - `sqs:StartMessageMoveTask`
   - `sqs:CancelMessageMoveTask`
   - `sqs:ListMessageMoveTasks`
   - `sqs:ListDeadLetterSourceQueues`
   - `sqs:ReceiveMessage`
   - `sqs:DeleteMessage`
   - `sqs:GetQueueAttributes`
   - デッドレターキューの `Resource` ARN (例えば、"arn:aws:sqs:*<DLQ_region>*:*<DLQ_accountId>*:*<DLQ_name>*")。
   - `sqs:SendMessage`
   - 送信先キューの `Resource` ARN (例：「arn:aws:sqs:*#DestQueue_region>*：*# DestQueue_accountId >*：*#DestQueue_name>*」) 。

   アクセスポリシーは以下のようになります。

```
{
"Version": "2012-10-17",
  "Statement": [
    {
"Effect": "Allow",
      "Action": [
        "sqs:StartMessageMoveTask",
        "sqs:CancelMessageMoveTask",
        "sqs:ListMessageMoveTasks",
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage",
        "sqs:GetQueueAttributes",
        "sqs:ListDeadLetterSourceQueues"
      ],
      "Resource": "arn:aws:sqs:<DLQ_region>:<DLQ_accountId>:<DLQ_name>"
    },
    {
"Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource":
 "arn:aws:sqs:<DestQueue_region>:<DestQueue_accountId>:<DestQueue_name>"
    }
  ]
}
```

CloudTrail Amazon SQS デッドレターキューリドライブの更新とアクセス許可の要件

2023 年 6 月 8 日、Amazon SQS は SDK と (CLI) の AWS デッドレターキュー AWS Command Line Interface (DLQ) リドライブを導入しました。この機能は、 AWS コンソールで既にサポートされている DLQ リドライブに追加されています。以前に AWS コンソールを使用してデッドレターキューメッセージをリドライブしたことがある場合は、次の変更の影響を受ける可能性があります。

- CloudTrail デッドレターキューリドライブのイベントの名前変更
- デッドレターキューリドライブのアクセス許可を更新

CloudTrail イベントの名前変更

2023 年 10 月 15 日、デッドレターキューリドライブの CloudTrail イベント名は Amazon SQS コンソールで変更されます。これらの CloudTrail イベントのアラームを設定している場合は、今すぐ更新する必要があります。DLQ リドライブの新しい CloudTrail イベント名は次のとおりです。

以前のイベント名	新しいイベント名
`CreateMoveTask`	`StartMessageMoveTask`
`CancelMoveTask`	`CancelMessageMoveTask`

更新されたアクセス許可

SDK および CLI リリースに含まれている Amazon SQS では、セキュリティのベストプラクティスに準拠するために DLQ リドライブのキューアクセス許可も更新されました。DLQs からメッセージをリドライブするには、次のキューのアクセス許可タイプを使用します。

1. アクションベースのアクセス許可 (DLQ API アクションの更新)
2. マネージド Amazon SQS ポリシーのアクセス許可
3. sqs:* ワイルドカードを使用するアクセス許可ポリシー

> **Important**
>
> SDK または CLI に DLQ リドライブを使用するには、上記のオプションのいずれかに一致する DLQ リドライブ許可ポリシーが必要です。

DLQ リドライブのキュー許可が上記のオプションのいずれとも一致しない場合は、2023 年 8 月 31 日までに許可を更新する必要があります。現在から 2023 年 8 月 31 日の間、アカウントは、以前に DLQ リドライブを使用したことがあるリージョンでのみ、 AWS コンソールを使用して設定したアクセス許可を使用してメッセージをリドライブできます。例えば、us-east-1 と eu-west-1 の両方に「アカウント A」があるとします。「アカウント A」は、2023 年 6 月 8 日より前に us-east-1 で AWS コンソールでメッセージをリドライブするために使用されていましたが、eu-west-1 では使用されていません。2023 年 6 月 8 日から 2023 年 8 月 31 日の間、「アカウント A の」ポリシーのアクセス許可が上記のオプションのいずれにも一致しない場合、コンソールでメッセージをリドライブするためにのみ使用でき、eu AWS -west-1 では使用できません。

> **Important**
>
> 2023 年 8 月 31 日以降に DLQ リドライブ許可がこれらのオプションのいずれにも一致しない場合、アカウントは AWS コンソールを使用して DLQ メッセージをリドライブできなくなります。
> ただし、2023 年 8 月に AWS コンソールで DLQ リドライブ機能を使用した場合、2023 年 10 月 15 日まで拡張機能を使用して、これらのオプションのいずれかに従って新しいアクセス許可を採用できます。
> 詳細については、「the section called “影響を受けるポリシーの特定”」を参照してください。

DLQ リドライブオプションごとのキュー許可の例を次に示します。サーバー側の暗号化 (SSE) キューを使用する場合、対応する AWS KMS キーのアクセス許可が必要です。

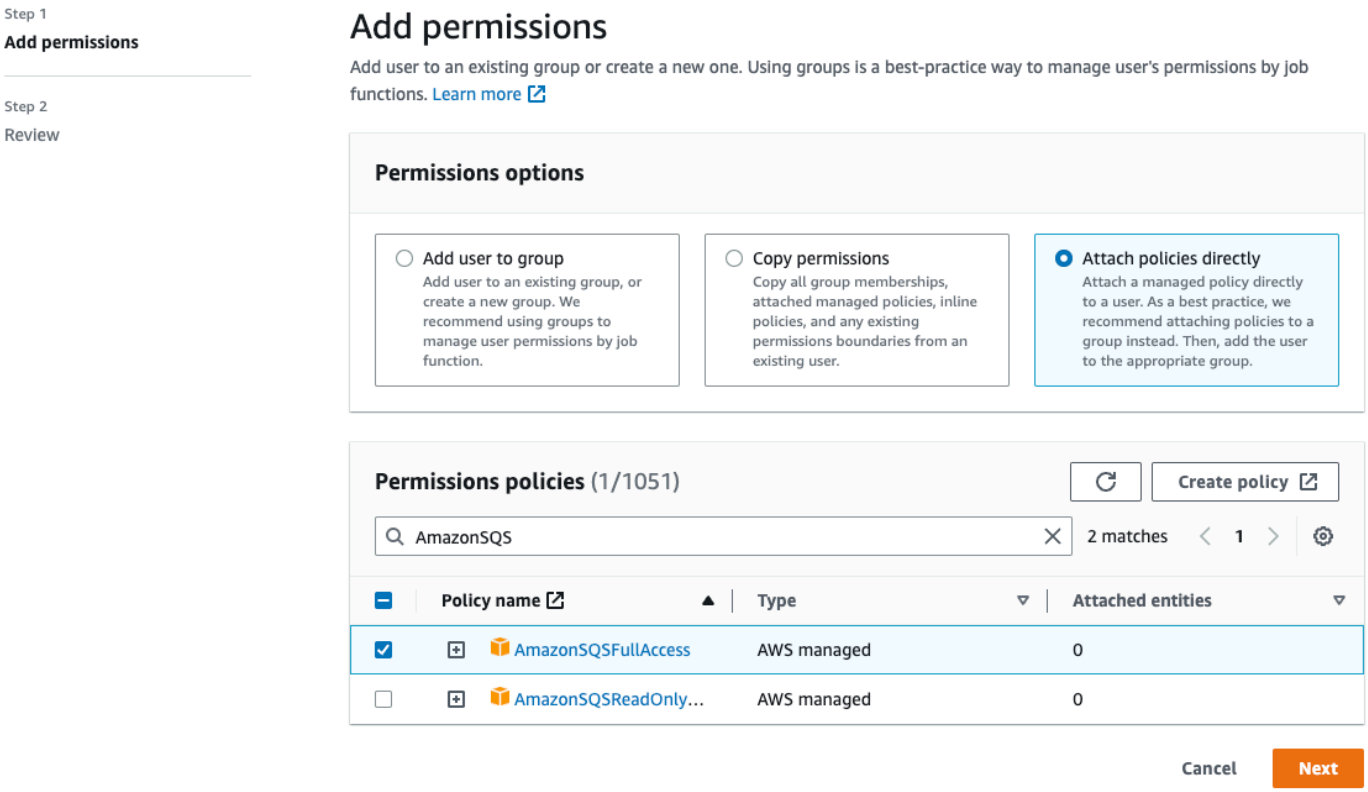
アクションベース

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage",
        "sqs:GetQueueAttributes",
        "sqs:StartMessageMoveTask",
        "sqs:ListMessageMoveTasks",
        "sqs:CancelMessageMoveTask"
      ],
      "Resource": "arn:aws:sqs:<DLQ_region>:<DLQ_accountId>:<DLQ_name>"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:SendMessage",
      "Resource":
 "arn:aws:sqs:<DestQueue_region>:<DestQueue_accountId>:<DestQueue_name>"
    }
  ]
}
```

マネージドポリシー

次の 管理ポリシーには、必要な更新アクセス許可が含まれています。

- AmazonSQSFullAccess – 開始、キャンセル、およびリストのデッドレターキューリドライブタスクが含まれます。
- AmazonSQSReadOnly Access – 読み取り専用アクセスを提供し、デッドレターキューのリドライブタスクのリストが含まれます。



sqs*ワイルドカードを使用するアクセス許可ポリシー

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

```
}
```

影響を受けるポリシーの特定

カスタマー管理ポリシー (CMPsを使用している場合は、 AWS CloudTrail と IAM を使用して、キューのアクセス許可の更新の影響を受けるポリシーを特定できます。

> **Note**
>
> AmazonSQSFullAccess と を使用している場合はAmazonSQSReadOnlyAccess、それ以上のアクションは必要ありません。

1. AWS CloudTrail コンソールにサインインします。
2. イベント履歴ページの属性の検索 で、ドロップダウンメニューを使用してイベント名 を選択します。次に、 を検索しますCreateMoveTask。
3. イベントを選択して、詳細ページを開きます。イベントレコードセクションで、ARN RoleNameから UserNameまたは userIdentity を取得します。
4. IAM コンソールにサインインします。
   - ユーザーの場合は、ユーザーを選択します。前のステップでUserName特定した を持つユーザーを選択します。
   - ロールには、ロールを選択します。前のステップでRoleName特定した を持つユーザーを検索します。
5. 詳細ページのアクセス許可セクションで、 のsqs:プレフィックスを持つポリシーを確認するかAction、 で Amazon SQS キューが定義されているポリシーを確認しますResource。

Amazon を使用してデッドレターキューのアラームを作成する CloudWatch

Amazon CloudWatch とメトリクス を使用して、デッドレターキューに移動されたメッセージに対してアラームを設定できますApproximateNumberOfMessagesVisible。詳細については、「Amazon SQS メトリクスの CloudWatch アラームの作成」を参照してください。メッセージがデッドレターキューに送信されたというアラートを受け取ったら、ポーリングを使用してメッセージを確認できます。

Amazon SQS のメッセージメタデータ

メッセージ属性を使用して、アプリケーション用のAmazon SQSメッセージにカスタムメタデータをアタッチすることができます。メッセージシステム属性を使用して、 AWS などの他の AWS X-Rayサービスのメタデータを保存できます。

トピック

- Amazon SQSメッセージ属性
- Amazon SQSメッセージシステム属性

Amazon SQSメッセージ属性

Amazon SQSでは、構造化メタデータ(タイムスタンプ、地理空間データ、署名、識別子など)をメッセージ属性を使用してメッセージに含めることができます。各メッセージには最大10個の属性を指定できます。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 (ただし、メッセージ本文とともに送信されます)。コンシューマーはメッセージ属性を使用して、最初にメッセージ本文を処理することなく、特定の方法でメッセージを処理できます。Amazon SQSコンソールを使用して属性とメッセージを送信する方法については、「属性を含むメッセージの送信」を参照してください。

> **Note**
>
> メッセージ属性とメッセージシステム属性を混同しないでください。メッセージ属性を使用してアプリケーションの Amazon SQSメッセージにカスタムメタデータをアタッチできますが、メッセージシステム属性を使用して、 などの他の AWS サービスのメタデータを保存できます AWS X-Ray。

トピック

- メッセージ属性コンポーネント
- メッセージ属性のデータ型
- メッセージ属性のMD5メッセージダイジェストの計算

メッセージ属性コンポーネント

> **Important**
>
> メッセージ属性のすべてのコンポーネントは、256KBというメッセージサイズの制限に含まれます。
> Name、Type、Value、およびメッセージ本文を空または Nullにすることはできません。

各メッセージ属性は、次のコンポーネント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。以下の制限が適用されます。
  - 最大256文字です
  - AWS.またはAmazon.(大文字小文字が異なるものを含む) でスタートすることはできません
  - 大文字と小文字を区別します
  - メッセージのすべての属性名で一意である必要があります
  - 先頭と末尾をピリオドにすることはできません
  - シーケンスにピリオドを含めることはできません
- タイプ-メッセージ属性のデータタイプ サポートされるタイプには String、Number、Binary などがあります。任意のデータ型のカスタム情報を追加することもできます。データタイプには、メッセージ本文と同じ制限があります (詳細については、 SendMessageの「Amazon Simple キューサービス API リファレンス」を参照してください)。また、以下の制限も適用されます。
  - 最大256文字です
  - 大文字と小文字を区別します
- 値-メッセージ属性値。Stringデータ型の場合、属性値の値にはメッセージ本文と同じ制限があります。

メッセージ属性のデータ型

メッセージ属性のデータ型により、対応するメッセージ属性値を処理する方法が Amazon SQS に指示されます。たとえば、タイプが Numberの場合、 Amazon SQSは数値を検証します。

Amazon SQS では、String、Number、Binary の各論理データタイプと、オプションの*.custom-data-type*形式のカスタムデータタイプラベルをサポートしています。

- 文字列-String属性はすべての有効なXML文字を使用してUnicodeテキストを保存できます。
- 数値-Number属性には、正または負の数値が保存できます。数値は最大38桁の精度で、10^-128から10^+126までの間とします。

> **Note**
>
> Amazon SQSでは先頭および末尾の0は削除されます。

- バイナリ-バイナリ属性には、圧縮データ、暗号化データ、イメージなど、すべてのバイナリデータが保存できます。
- カスタム-カスタムデータ型を作成するには、任意のデータ型にカスタム型ラベルを追加します。例:
  - `Number.byte`、`Number.short`、`Number.int`、および `Number.float` は、数値型の区別ができます。
  - `Binary.gif` および `Binary.png` は、ファイルタイプの区別ができます。

> **Note**
>
> Amazon SQSが、追加されたデータを解釈、検証、または使用することはありません。カスタム型ラベルには、メッセージ本文と同じ制限があります。

メッセージ属性のMD5メッセージダイジェストの計算

を使用する場合は AWS SDK for Java、このセクションをスキップできます。SDK for Javaの `MessageMD5ChecksumHandler` クラスでは、Amazon SQS メッセージ属性の MD5 メッセージダイジェストがサポートされています。

クエリ API または Amazon SQS メッセージ属性の MD5 メッセージダイジェストをサポートしていない AWS SDKs のいずれかを使用する場合は、次のガイドラインを使用して MD5 メッセージダイジェスト計算を実行する必要があります。 Amazon SQS

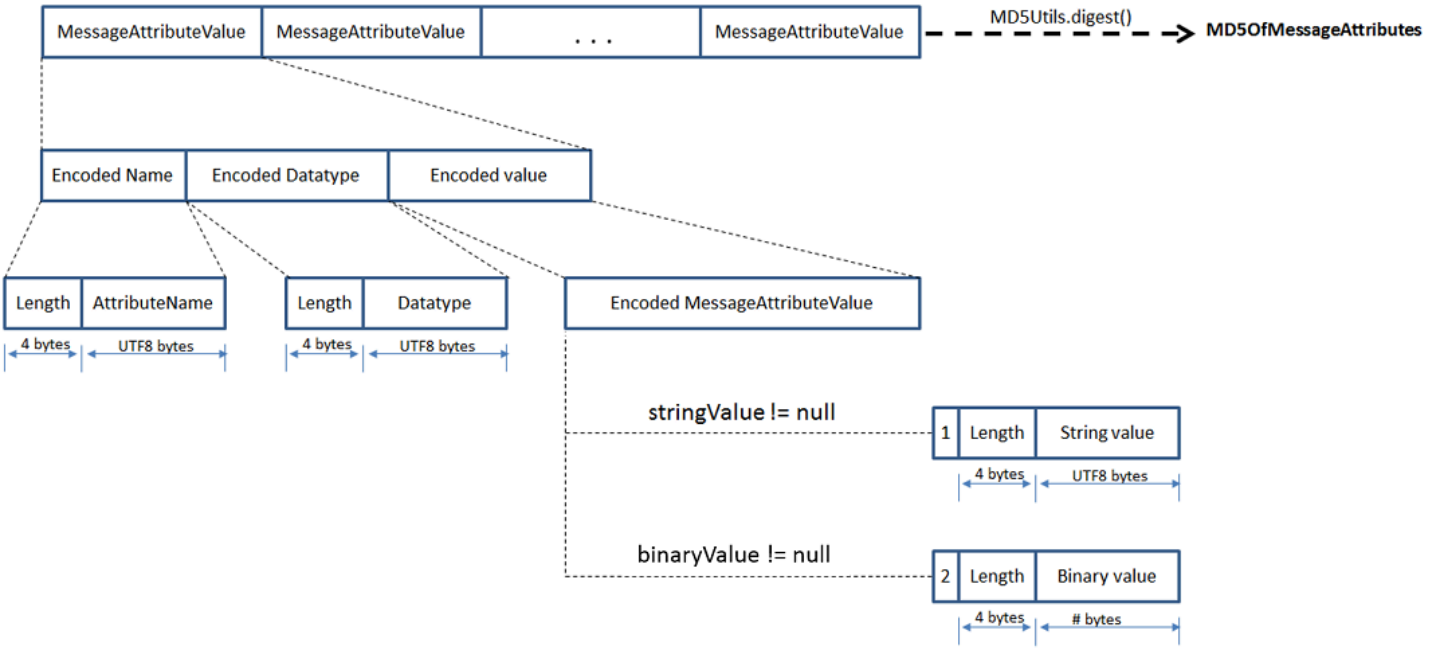
> **Note**
>
> MD5メッセージダイジェストの計算には、常にカスタムデータタイプサフィックスを含めてください。

概要

MD5メッセージダイジェスト計算アルゴリズムの概要を以下に示します。

1. すべてのメッセージ属性を名前で昇順にソートします。
2. 各属性 (Name、Type、Value) の個々の部分をバッファにエンコードします。
3. バッファ全体のメッセージダイジェストを計算します。

次の図は、1つのメッセージ属性のMD5メッセージダイジェストをエンコードする方法を示しています。



単一のAmazon SQSメッセージ属性をエンコードするには

1. 名前をエンコードします: 名前の長さ (4バイト) およびUTF-8バイト。
2. データ型をエンコードします:データ型の長さ (4バイト) および UTF-8バイト。
3. 値 (1バイト) の転送型 (StringまたはBinary) をエンコードします。

> **Note**
>
> 論理データ型 StringおよびNumberでは、String転送型が使用されます。
> 論理データ型 Binaryでは、Binary転送型が使用されます。

a. String転送型の場合、1をエンコードします。

b. Binary転送型の場合、2をエンコードします。

4. 属性値をエンコードします。

a. String転送型の場合、属性値をエンコードします:値の長さ(4バイト) + UTF-8バイト。

b. Binary転送型の場合、属性値をエンコードします:値の長さ(4バイト) + rawバイト。

Amazon SQSメッセージシステム属性

メッセージ属性を使用してアプリケーションの Amazon SQS メッセージにカスタムメタデータをアタッチできますが、メッセージシステム属性 AWS を使用して、 AWS X-Rayなどの他のサービスのメタデータを保存できます。詳細については、Amazon Simple キューサービス API リファレンス「SendMessageBatchAPI アクションMessageSystemAttributeのリクエストパラメーターSendMessage、API アクションAWSTraceHeaderの属性ReceiveMessage、およびMessageSystemAttributeValueのデータタイプ。」を参照してください。

メッセージシステム属性は、メッセージ属性とまったく同じ構造ですが、次の例外があります。

- 現在、サポートされているメッセージシステム属性はAWSTraceHeaderのみです。そのタイプはStringで、値は正しくフォーマットされた AWS X-Ray トレースヘッダー文字列である必要があります。
- メッセージシステム属性のサイズは、メッセージの合計サイズに対してはカウントされません。

Amazon SQSメッセージの処理に必要なリソース

キューに配置されたメッセージを処理するために必要なリソースが予測しやすいように、Amazon SQSではキュー内の遅延、可視、不可視メッセージの概数を判断できます。可視性の詳細については、「Amazon SQS可視性タイムアウト」を参照してください。

> **Note**
>
> スタンダードキューの場合、Amazon SQSの分散アーキテクチャのため、結果は概算になります。ほとんどの場合、カウントはキュー内の実際のメッセージ数に近い数値になります。FIFOキューの場合、結果は正確です。

以下の表に、GetQueueAttributesアクションで使用する属性名の一覧を示します:

タスク	属性名
キューから取得可能なメッセージのおおよその数を取得します。	ApproximateNumberOfMessages Visible
キュー内の、遅延が発生したためにすぐに読み取ることができないメッセージのおおよその数を取得します。これは、キューが遅延キューとして設定されている場合、またはメッセージが遅延パラメータとともに送信された場合に発生することがあります。	ApproximateNumberOfMessages Delayed
処理中のメッセージのおおよその数を取得します。メッセージがクライアントに送信されたが、まだ削除されていない場合、または表示期限に達していない場合、メッセージは処理中とみなされます。	ApproximateNumberOfMessages NotVisible

キューのページ割りの一覧表示

listQueuesそしてlistDeadLetterQueuesAPIメソッドは、オプションのページ割りコントロールをサポートします。デフォルトでは、これらのAPIメソッドはレスポンスメッセージで最大1000通のキューを返信します。MaxResultsパラメータを指定すると、各レスポンスで返信される結果が少なくなります。

listQueues またはlistDeadLetterQueuesリクエストにパラメータMaxResultsを設定して、レスポンスで返信される結果の最大数を指定します。設定しない場合MaxResultsの場合、レスポンスには最大1,000件の結果が含まれ、NextToken応答の値が NULL となります。

さらに表示する結果がある場合は、 MaxResultsを設定するとレスポンスにNextTokenの値が含まれます。結果の次のページを受け取るには、listQueues に対する次のリクエストで NextToken をパラメータとして使用します。レスポンスでNextTokenの値がnullの場合、表示できる追加の結果はありません。

Amazon SQSコスト配分タグ

コスト配分のために Amazon SQS キューを整理して識別するには、キューの目的、所有者、または環境を識別するメタデータタグを追加できます。これは、キューが多い場合に特に便利です。Amazon SQSコンソールを使用してタグを設定するには、「the section called “キューにタグを設定する”」を参照してください。

コスト配分タグを使用して、独自のコスト構造を反映するように AWS 請求書を整理できます。これを行うには、サインアップして、タグのキーと値を含めるための AWS アカウント 請求書を取得します。詳細については、AWS Billing ユーザーガイドの月別コスト配分レポートの設定を参照してください。

各タグは、ユーザー定義のキー-バリュー (1つのキーと1つの値) で構成されます。たとえば、次のようなタグをキューに付けると、本番稼働用とテスト用のキューを簡単に識別できます。

キュー	キー	値
MyQueueA	QueueType	Production
MyQueueB	QueueType	Testing

> **Note**
>
> キュータグを使用する場合は、以下のガイドラインに注意してください。
>
> - 1つのキューに50 個以上のタグを追加することはお勧めしません。タグは、UTF-8 の Unicode 文字をサポートします。
> - タグには意味論的な意味がありません。Amazon SQSはタグを文字列として解釈します。
> - タグは、大文字と小文字が区別します。
> - 既存のタグと同じキーを持つ新しいタグは、既存のタグを上書きします。
> - タグ付けアクションは、 あたり 30 TPS に制限されています AWS アカウント。アプリケーションが高スループットが必要な場合、リクエストを送信します。 。
>
> タグの制限事項一覧については、「クォータ」を参照してください。

Amazon SQSショートポーリングとロングポーリング

Amazon SQSは、キューからメッセージを受信するためのショートポーリングオプションとロングポーリングオプションを提供します。次の 2 つのポーリングオプションから選択する場合は、応答性とコスト効率に関するアプリケーションの要件を考慮してください。

- ショートポーリング (デフォルト) — `ReceiveMessage`リクエストは、サーバーサブセット (加重ランダムディストリビューションに基づく) にクエリを実行して使用可能なメッセージを検索し、メッセージが見つからない場合でも即時応答を送信します。
- ロングポーリング – すべてのサーバーにメッセージがないか`ReceiveMessage`クエリし、少なくとも 1 つのメッセージが使用可能になると、指定された最大値まで応答を送信します。空のレスポンスは、ポーリング待機時間が経過した場合にのみ送信されます。このオプションを使用すると、空のレスポンスの数が減少し、コストが削減される可能性があります。

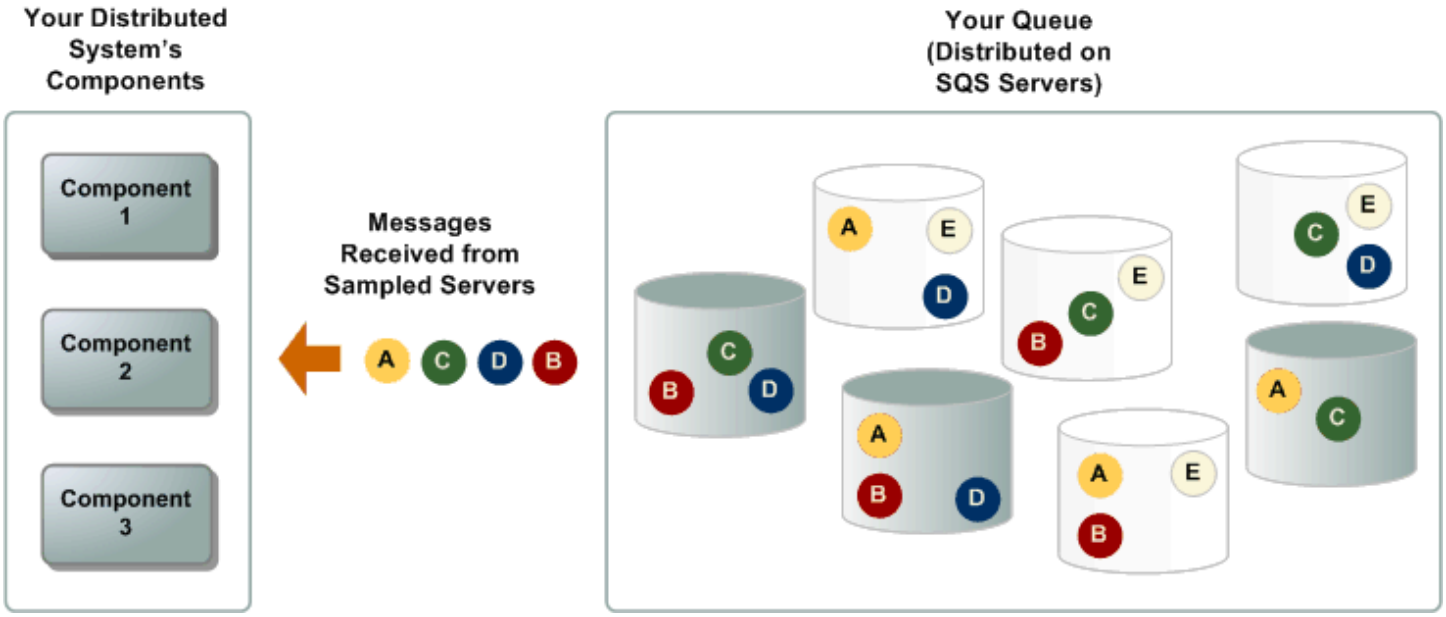
次のセクションでは、ショートポーリングとロングポーリングの詳細について説明します。

トピック

ショートポーリングを使用したメッセージの処理

ショートポーリングを使用してキュー (FIFO または標準) からのメッセージを使用する場合、Amazon SQS はサーバーのサブセットをサンプリングし (加重ランダムディストリビューションに基づく) 、それらのサーバーからのメッセージのみを返します。したがって、ある `ReceiveMessage` リクエストによってすべてのメッセージが返されないことがあります。ただし、キューにあるメッセージが1,000未満の場合、後続のリクエストではメッセージが返されます。キューから処理し続けた場合、Amazon SQSによりすべてのサーバーがサンプリングされ、すべてのメッセージを受信します。

次の図は、システムコンポーネントの1 が受信リクエストを生成すると スタンダードキューからメッセージが返されるショートポーリングの動作を示しています。Amazon SQSは、複数のサーバー (灰色) をサンプリングし、それらのサーバーからメッセージ A、C、D、および B を返信します。メッセージ E はこのリクエストに返されませんが、後続のリクエストには返されます。



ロングポーリングを使用したメッセージの使用

待ち時間がいつになるかReceiveMessageAPI アクションが0より大きい場合ロングポーリングが有効です。長いポーリングの最大待機時間は20秒です。ロングポーリングは、空のレスポンスと偽の空の応答の数を排除することで、Amazon SQS を使用するためのコストを削減します。（ReceiveMessageリクエストメッセージが利用可能であるが応答に含まれていない場合）。Amazon SQSコンソールを使用して、新しいキューまたは既存のキューのロングポーリングを有効にする方法については、Amazon SQS コンソールを使用したキューパラメータの設定を参照してください。ベストプラクティスについては、ロングポーリングのセットアップを参照してください。

ロングポーリングには次の利点があります。

- レスポンスの送信前にメッセージがキューで使用可能になるまで Amazon SQSが待機できるように、空のレスポンス数を削減します。接続がタイムアウトしない限り、ReceiveMessageリクエストに対するレスポンスに、使用可能なメッセージが少なくとも1つ、最大でReceiveMessageアクションに指定されたメッセージ数まで含まれます。まれに、キューにまだメッセージが含まれている場合でも、空の応答が受信されることがあります、特に、ReceiveMessageWaitTimeSecondsパラメータに低い値を指定した場合。
- サブセットではなく、すべての Amazon SQSサーバーにクエリを実行して、偽の空のレスポンスを減らします。
- 利用可能になるとすぐにメッセージを返します。

キューが空であることを確認する方法については、Amazon SQS キューが空であることを確認するを参照してください。

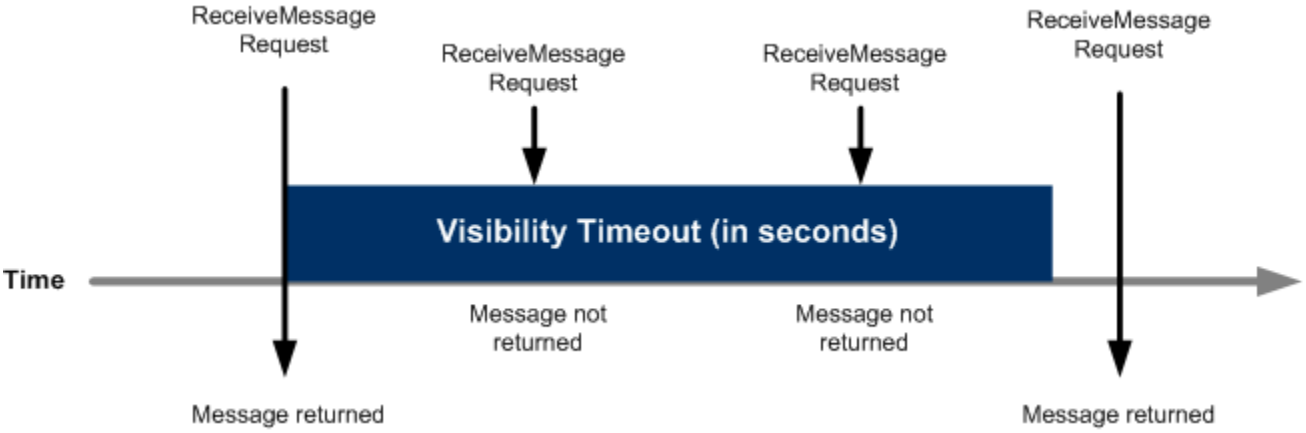
ロングポーリングとショートポーリングの違い

ショートポーリングは、`WaitTimeSeconds`リクエストの`ReceiveMessage`パラメータを次の2通りの方法で`0`に設定すると行われます。

- `ReceiveMessage`コールは `WaitTimeSeconds` を `0` に設定します。
- `ReceiveMessage` コールでは `WaitTimeSeconds` を設定しませんが、キューの属性 `ReceiveMessageWaitTimeSeconds` は `0` に設定されます。

Amazon SQS可視性タイムアウト

コンシューマーがキューからメッセージを受信して処理しても、そのメッセージはキューに保留されたままです。Amazon SQSでは、メッセージが自動的に削除されません。Amazon SQSは分散システムであり、接続の問題やコンシューマーアプリケーションの問題などが原因で、コンシューマーが実際にメッセージを受信するという保証がありません。そのため、コンシューマーはメッセージを受信して処理した後、キューからメッセージを削除する必要があります。



メッセージが受信された直後は、メッセージはキューに保留されたままです。他のコンシューマーがメッセージを再度処理できないように、Amazon SQS は可視性タイムアウト を設定します。これは、Amazon SQS がすべてのコンシューマーがメッセージの受信と処理を禁止している期間です。デフォルトの可視性タイムアウトは30秒です。最小は0秒です。最大は12時間です。コンソールを使用したキューの可視性タイムアウトの設定については、「Amazon SQS コンソールを使用したキューパラメータの設定」を参照してください。

> **Note**
>
> スタンダードキューの場合は、可視性タイムアウトはメッセージを2回受信しない保証にはなりません。詳細については、「t-least-once 配信」を参照してください。
> FIFOキューでは、プロデューサまたはコンシューマが複数の再試行を試みることができます。
>
> - プロデューサが障害を検出したSendMessageアクションがある場合、同じメッセージ重複除外 IDを使用して、必要な回数だけ送信を再試行できます。重複除外間隔の期限が切れる前にプロデューサが少なくとも1つの確認応答を受信すると仮定すると、複数回の再試行はメッセージの順序に影響せず、重複も発生しません。
> - コンシューマが障害を検出した場合ReceiveMessageアクションの場合、同じ受信リクエスト試行 IDを使用して、必要な回数だけ再試行できます。可視性タイムアウトの期限が切れる前にコンシューマが少なくとも1つの確認応答を受信すると仮定すると、複数回の再試行はメッセージの順序に影響しません。
> - メッセージグループ IDがあるメッセージを受信した場合、メッセージを削除するか、表示されない限り、同じメッセージグループ IDのメッセージはそれ以上返信されません。

トピック

インフライトメッセージ

Amazon SQS メッセージには、次の 3 つのベーシックな状態があります。

1. プロデューサーからキューに送信されます。
2. コンシューマがキューから受信しました。
3. キューから削除されました。

メッセージは、次のものと見なされます。保存されたプロデューサからキューに送信され、コンシューマによってキューからまだ受信されていない (つまり、1と2の間の状態)。保存されるメッ

セージの数にクォータはありません。メッセージは、次のものと見なされます。保留中コンシューマがキューから受信したが、キューから削除されていない (つまり、ステート2と3の間の状態) 。インフライトメッセージの数にはクォータがあります。

> **Important**
>
> インフライトメッセージに適用されるクォータは、保存されているメッセージの数に制限はありません。

ほとんどのスタンダードキューでは (キュートラフィックとメッセージバックログにより異なります)、最大約 120,000 通の処理中のメッセージ (コンシューマーからキューを受信し、キューからまだ削除されていないもの) が存在する可能性があります。ショートポーリングを使用している場合、このクォータに達すると、Amazon SQS は`OverLimit`エラーメッセージをし返信します。ロングポーリングを使用している場合、 Amazon SQSはエラーメッセージを返しません。クォータに到達しないように、処理されたメッセージはキューから削除する必要があります。メッセージの処理に使用するキューの数を増やすこともできます。クォータの引き上げをリクエストするには、サポートリクエストを送信します。

FIFO キューでは、最大 20,000 通 の処理中メッセージ (コンシューマがキューから受信して、キューから削除されていないもの) が存在する可能性があります。このクォータに届いた場合、Amazon SQSはエラーメッセージを返しません。

> **Important**
>
> FIFO キューを使用する場合、可視性タイムアウトウィンドウ外でリクエストが受信されると、`DeleteMessage`オペレーションは失敗します。可視性タイムアウトが 0 秒の場合、メッセージは送信されたのと同じミリ秒以内に削除する必要があります。そうしないと、中止されたと見なされます。これにより、`MaxNumberOfMessages`パラメータが 1 Amazon SQS は`ReceiveMessage`オペレーションへの同じレスポンスに重複メッセージを含める可能性があります。詳細については、Amazon SQSFIFO API の仕組み」を参照してください。

可視性タイムアウトの設定

可視性タイムアウトはAmazon SQSがメッセージを返した時点で開始します。タイムアウト時間内に、コンシューマーはメッセージを処理して削除します。ただし、コンシューマーでメッセージを削除する前に障害が発生して、`DeleteMessage`アクションが呼び出されないまま可視性タイムアウト

の期限が切れると、そのメッセージは他のコンシューマーに見えるようになり、再度受信されます。メッセージを一度だけ受信する必要がある場合、コンシューマーは可視性タイムアウトの時間内にメッセージを削除する必要があります。

すべてのAmazon SQSキューの可視性タイムアウトはデフォルトで 30秒に設定されています。この設定はキュー全体で変更できます。通常、可視性タイムアウトは、アプリケーションがキューのメッセージを処理して削除するまでの最大所要時間に設定します。メッセージを受信したら、キュー全体のタイムアウトを変更しなくても、返されるメッセージに特別な可視性タイムアウトを設定することもできます。詳細については、「タイムリーな方法でのメッセージの処理」セクションのベストプラクティスを参照してください。

メッセージの処理に要する時間がわからない場合は、ハートビートコンシューマプロセスの場合:初期可視性タイムアウト (2 分など) を指定し、コンシューマがメッセージで作業している限り、可視性タイムアウトを1分ごとに2分延長します。

> **Important**
>
> 最大可視性タイムアウトは、ReceiveMessageAmazon SQS がメッセージを受信してから 12 時間です。可視性タイムアウトを延長しても、最大 12 時間はリセットされません。
> さらに、ReceiveMessageリクエストがタイマーを開始してから 12 時間 (43,200 秒など) は、個々のメッセージのタイムアウトを設定できない場合があります。
> 例えば、メッセージを受信し、すぐに 43,200 秒VisibilityTimeoutに等しいのChangeMessageVisibility呼び出しを送信して最大 12 時間を設定すると、失敗する可能性があります。ただし、経由でメッセージをリクエストReceiveMessageしてから可視性タイムアウトを更新するまでに大幅な遅延がない限り、43,195 秒の値を使用すると機能します。コンシューマーが12時間以上必要とする場合は、ステップ機能 の使用を検討してください。

メッセージの可視性タイムアウトの変更

キューからメッセージを受信し、処理を開始したときに、キューの可視性タイムアウトが十分でない場合があります (たとえば、メッセージを処理して削除する必要がある場合)。ChangeMessageVisibility アクションを使用して新しいタイムアウト値を指定すると、メッセージの可視性を短縮または拡張することができます。

たとえば、キューのデフォルトのタイムアウトが60秒であり、メッセージを受信してから15秒が経過したときに、ChangeMessageVisibilityを10秒に設定したVisibilityTimeoutコールを送

信する場合、この10秒はChangeMessageVisibilityコールを行った時点からカウントが開始されます。したがって、最初に可視性タイムアウトを変更してから10秒 (合計25秒) 経過した後に可視性タイムアウトを変更しようとしたり、そのメッセージを削除しようとすると、エラーが発生する可能性があります。

> **Note**
>
> 新しいタイムアウト期間は、ChangeMessageVisibilityアクションを呼び出した時間から有効になります。さらに、新しいタイムアウト期間はメッセージの特定の受信にのみ適用されます。ChangeMessageVisibilityは、メッセージの以降の受信や以降のキューには影響しません。

メッセージの可視性タイムアウトの終了

キューからメッセージを受信すると、そのメッセージを実際には処理して削除する必要がないとわかる場合があります。Amazon SQSでは、特定のメッセージの可視性タイムアウトを終了できます。その場合、システムの他のコンポーネントからメッセージがすぐに見えるようになり、処理できるようになります。

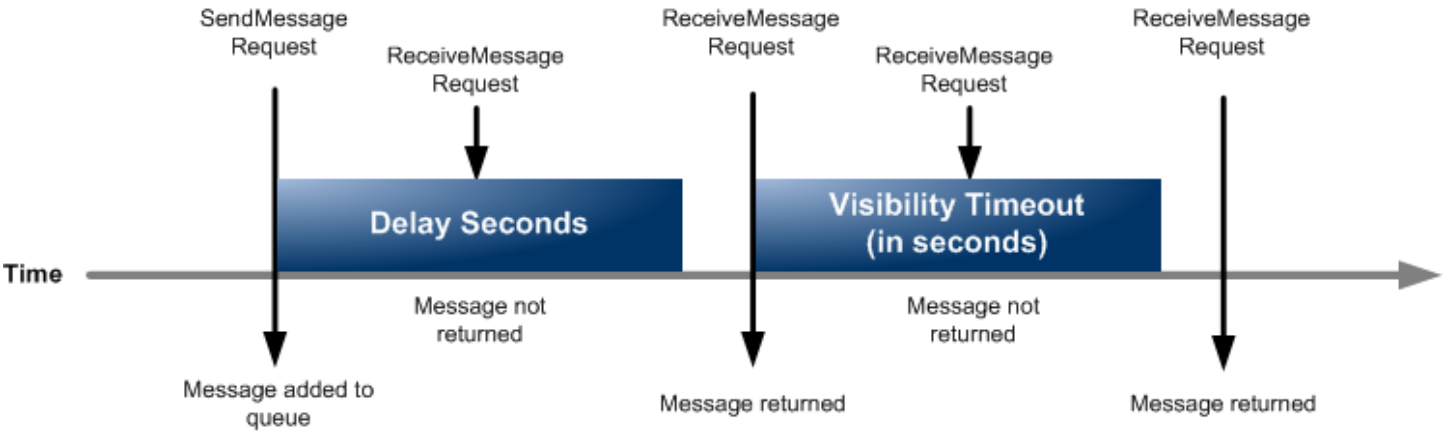
ReceiveMessageを呼び出した後にメッセージの可視性タイムアウトを終了するにはChangeMessageVisibility、を呼び出します。VisibilityTimeoutと0秒に設定します。

Amazon SQS遅延キュー

遅延キューは、例えば、コンシューマアプリケーションがメッセージ処理にさらに時間を必要な場合、コンシューマへ新しいメッセージの配信を数秒延期することができます。遅延キューを作成した場合、そのキューに送信したすべてのメッセージは遅延期間中にコンシューマーに表示されません。キューのデフォルト (最小) 遅延は0秒です。最大数は15分です。コンソールを使用して遅延キューを設定する方法の詳細については、「Amazon SQS コンソールを使用したキューパラメータの設定」を参照してください。

> **Note**
>
> スタンダードキューの場合は、キューごとの遅延設定には遡及性はありません-設定を変更しても、既にキューにあるメッセージの遅延には影響しません。
> FIFOキューの場合は、キューごとの遅延設定に遡及性があります-設定を変更すると、既にキューにあるメッセージの遅延に影響します。

遅延キューでは、コンシューマーはメッセージを一定の時間使用できなくなるため、可視性タイムアウトと似ています。両者の違いは、遅延キューの場合はメッセージが最初にキューに追加されたときに非表示になるのに対し、可視性タイムアウトの場合は、メッセージがキューから処理された後のみ非表示になるという点です。次の図は、遅延キューと可視性タイムアウトの関係を示しています。



キュー全体に対してではなく、各メッセージに遅延の秒数を設定するには、メッセージタイマーを使用して、Amazon SQSが遅延キューのDelaySeconds値ではなく、メッセージタイマーの値をDelaySeconds使用できるようにします。

Amazon SQS一時キュー

一時キューは、リクエストとレスポンスのような一般的なメッセージパターンを使用する場合に、開発時間と開発コストを削減するのに役立ちます。一時キュークライアントを使用して、高スループットでコスト効率の高いアプリケーション管理の一時キューを作成します。

クライアントは複数のマッピングを行います。一時キュー— 特定のプロセスに対してオンデマンドで作成されるアプリケーション管理キューを、単一の Amazon SQSキューに自動的に作成します。これにより、各一時キューへのトラフィックが少ないときのアプリケーションのAPI呼び出しが少なくなり、スループットを高めることができます。一時キューが使用されなくなると、クライアントを使用する一部のプロセスが正常にシャットダウンされない場合でも、クライアントは一時キューを自動的にクリーンアップします。

一時キューの利点を以下に示します:

- これらは、特定のスレッドまたはプロセスの軽量通信チャネルとして機能します。
- 追加のコストが発生することなく作成および削除できます。
- これらは静的 (通常)Amazon SQSキューとAPI互換です。つまり、メッセージを送受信する既存のコードは、仮想キューとの間でメッセージを送受信できます。

トピック

仮想キュー

仮想キューは、Temporary Queue Clientが作成するローカルデータ構造です。仮想キューを使用すると、トラフィックの少ない複数の宛先を単一のAmazon SQSキューに結合できます。ベストプラクティスについては、仮想キューでの同じメッセージグループ ID の再使用を避けるを参照してください。

> **Note**
>
> - 仮想キューを作成すると、コンシューマーがメッセージを受信するための一時的なデータ構造のみが作成されます。仮想キューはAmazon SQSへのAPI呼び出しを行わないため、仮想キューにはコストはかかりません。
> - TPSクォータは、単一ホストキューのすべての仮想キューに適用されます。詳細については、「Amazon SQS メッセージクォータ」を参照してください。

AmazonSQSVirtualQueuesClientラッパークラスは、仮想キューに関連する属性のサポートを追加します。仮想キューを作成するには、 CreateQueue 属性を使用してHostQueueURLAPI アクションを呼び出す必要があります。この属性は、仮想キューをホストする既存のキューを指定します。

仮想キューのURLは次の形式になります。

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue#MyVirtualQueueName
```

プロデューサーが仮想キュー URLでSendMessageまたはSendMessageBatch API アクションを呼び出すと、Temporary Queue Clientでは次の処理が実行されます:

1. 仮想キュー名を抽出します。
2. 追加のメッセージ属性として仮想キュー名にアタッチします。
3. ホストのキューにメッセージを送信します。

プロデューサーがメッセージを送信する間、バックグラウンドスレッドはホストキューをポーリングし、対応するメッセージ属性に従って受信メッセージを仮想キューに送信します。

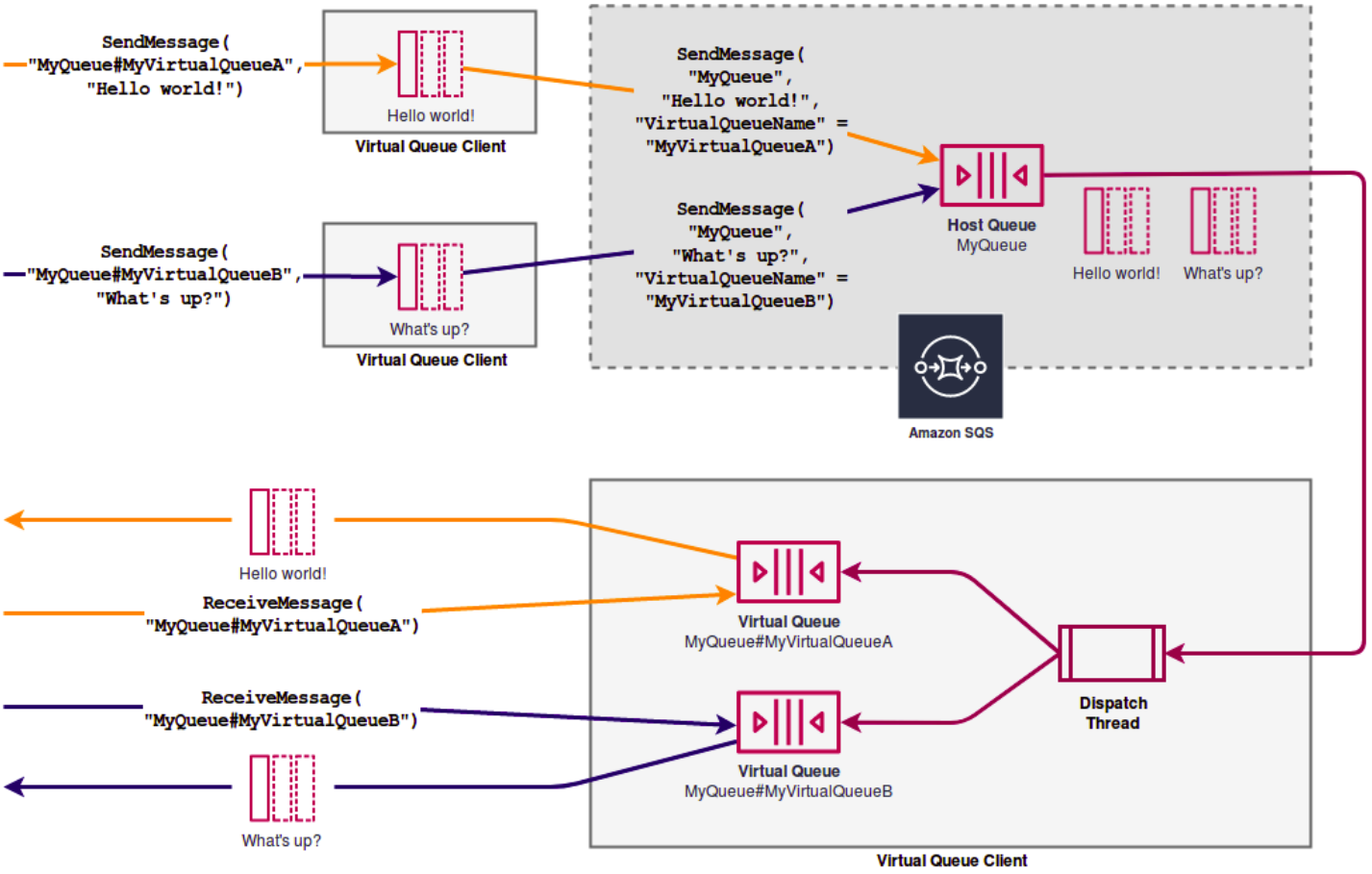
コンシューマが仮想キュー URL で`ReceiveMessage`API アクションを呼び出す間、Temporary Queue Clientは、バックグラウンドスレッドが仮想キューにメッセージを送信するまで、ローカルで呼び出しをブロックします。(このプロセスは、バッファ付き非同期クライアントでのメッセージのプリフェッチに似ています。1 つの APIアクションで最大 10 個の仮想キューにメッセージを提供できます。) 仮想キューを削除すると、Amazon SQS自体を呼び出すことなく、クライアント側のリソースが削除されます。

`AmazonSQSTemporaryQueuesClient`クラスは、作成したすべてのキューを一時キューに自動的に変換します。また、オンデマンドで、同じキュー属性を持つホストキューを自動的に作成します。これらのキューの名前は、一時キューとして識別される共通の構成可能なプレフィックス(デフォルトでは`__RequesterClientQueues__`) を共有します。これにより、クライアントは、キューを作成および削除する既存のコードを最適化するドロップイン置換として機能できるようになります。クライアントには、キュー間の双方向通信を可能にする `AmazonSQSRequester` および `AmazonSQSResponder` インターフェイスも含まれています。

リクエスト–レスポンスメッセージングパターン(仮想キュー)

一時キューの最も一般的な使用例は、リクエスト-レスポンスメッセージングパターンです。このパターンでは、要求者が各レスポンスメッセージを受信するための一時キューを作成します。レスポンスメッセージごとにAmazon SQSキューが作成されないように、Temporary Queue Clientを使用すると、Amazon SQS API呼び出しを行わずに複数の一時キューを作成および削除できます。詳細については、「リクエストと応答システムの実装」を参照してください。

このパターンを使用した一般的な構成を次の図に示します。



シナリオ例:ログインリクエストの処理

次のシナリオ例では、AmazonSQSRequesterとAmazonSQSResponderインターフェイスを使用して、ユーザーのログインリクエストを処理する方法を示しています。

クライアント側

```
public class LoginClient {

    // Specify the Amazon SQS queue to which to send requests.
    private final String requestQueueUrl;

    // Use the AmazonSQSRequester interface to create
    // a temporary queue for each response.
    private final AmazonSQSRequester sqsRequester =
            AmazonSQSRequesterClientBuilder.defaultClient();

    LoginClient(String requestQueueUrl) {
        this.requestQueueUrl = requestQueueUrl;
```

```
    }

    // Send a login request.
    public String login(String body) throws TimeoutException {
        SendMessageRequest request = new SendMessageRequest()
                .withMessageBody(body)
                .withQueueUrl(requestQueueUrl);

        // If no response is received, in 20 seconds,
        // trigger the TimeoutException.
        Message reply = sqsRequester.sendMessageAndGetResponse(request,
                20, TimeUnit.SECONDS);

        return reply.getBody();
    }
}
```

ログインリクエストを送信すると、次の処理が実行されます。

1. 一時テーブルを作成します。
2. 一時キューの URLを属性としてメッセージに添付します。
3. メッセージを送信します。
4. 一時キューからのレスポンスを受け取ります。
5. 一時キューを削除します。
6. レスポンスを返します。

サーバー側

次の例では、構築時に、キューをポーリングしてすべてのメッセージに対してhandleLoginRequest()メソッドを呼び出すスレッドが作成されることを想定しています。さらに、doLogin() メソッドを想定しています。

```
public class LoginServer {

    // Specify the Amazon SQS queue to poll for login requests.
    private final String requestQueueUrl;

    // Use the AmazonSQSResponder interface to take care
    // of sending responses to the correct response destination.
    private final AmazonSQSResponder sqsResponder =
```

```
            AmazonSQSResponderClientBuilder.defaultClient();

    LoginServer(String requestQueueUrl) {
        this.requestQueueUrl = requestQueueUrl;
    }

    // Process login requests from the client.
    public void handleLoginRequest(Message message) {

        // Process the login and return a serialized result.
        String response = doLogin(message.getBody());

        // Extract the URL of the temporary queue from the message attribute
        // and send the response to the temporary queue.
        sqsResponder.sendResponseMessage(MessageContent.fromMessage(message),
                new MessageContent(response));
    }
}
```

キューをクリーンアップする

Amazon SQS. が仮想キューで使用されているメモリ内リソースを確実に回収するために、アプリケーションで Temporary Queue Client が不要になったら、shutdown( )メソッドを呼び出す必要があります。shutdown( )インターフェイスのAmazonSQSRequesterメソッドを使用することもできます。

Temporary Queue Client孤立したホストキューを削除する方法も提供します。一定期間 (デフォルトでは5分間) にわたって API 呼び出しを受信する各キューについて、クライアントはTagQueueAPIアクションを使用して、使用中のキューにタグを付けます。

> **Note**
>
> キューで実行されたAPIアクションは、メッセージを返さないReceiveMessage アクションを含め、キューをアイドル以外としてマークします。

バックグラウンドスレッドは ListQueues および ListTags APIアクションを使用して、構成されたプレフィックスを持つすべてのキューをチェックし、少なくとも5分間タグ付けされていないキューを削除します。このようにして、1つのクライアントが正常にシャットダウンしない場合、他のアクティブなクライアントはその後クリーンアップします。作業の重複を減らすために、同じプ

レフィックスを持つすべてのクライアントは、プレフィックスにちなんで命名された共有内部作業キューを介して通信します。

Amazon SQSメッセージタイマー

メッセージタイマーを使用すると、キューに追加されたメッセージの初期非表示期間を特定できます。たとえば、45秒のタイマーでメッセージを送信すると、キューの最初の45秒間はコンシューマーにはメッセージが表示されません。メッセージのデフォルト（最小）遅延は0秒です。最大数は15分です。コンソール を使用してタイマーでメッセージを送信する方法については、「メッセージの送信」を参照してください。

> **Note**
>
> FIFOキューは、個々のメッセージのタイマーをサポートしていません。

個々のメッセージではなくキュー全体に対して遅延の秒数を設定するには、遅延キューを使用します。個々のメッセージのメッセージタイマー設定は、 Amazon SQS遅延キューのすべてのDelaySeconds 値よりも優先されます。

Amazon SQS コンソールから Amazon EventBridge Pipes にアクセスする Amazon SQS

Amazon EventBridge Pipes はソースをターゲットに接続します。Pipes は、サポートされているソースとターゲットの統合を目的として point-to-pointおり、高度な変換とエンリッチメントをサポートしています。 EventBridge Pipes は、Amazon SQS キューを Step Functions、Amazon SQS、API Gateway などの AWS サービス、および Salesforce などのサードパーティーの Software as a Service (SaaS) アプリケーションに接続するための、高度にスケーラブルな方法を提供します。

パイプをセットアップするには、ソースを選択し、オプションのフィルタリングを追加し、オプションのエンリッチメントを定義し、イベントデータのターゲットを選択します。

Amazon SQS キューの詳細ページでは、そのキューをソースとして使用しているパイプを表示できます。そこから、次の事柄も実行できます。

- EventBridge コンソールを起動してパイプの詳細を表示します。

- EventBridge コンソールを起動して、キューをソースとする新しいパイプを作成します。

Amazon SQS キューをパイプソースとして設定する方法の詳細については、「Amazon ユーザーガイド」の「Amazon Amazon SQS キューをソースとして設定する EventBridge 」を参照してください。 EventBridge パイプ全般の詳細については、EventBridge 「パイプ」を参照してください。

特定の Amazon SQS キューの EventBridge パイプにアクセスするには

1. Amazon SQSコンソールの キュー ページを開きます。
2. キューを選択します。
3. キューの詳細ページで、EventBridge パイプタブを選択します。

   EventBridge Pipes タブには、選択したキューをソースとして使用するように現在設定されているパイプのリストが含まれます。

   - パイプ名
   - 現在の状態
   - パイプのターゲット
   - パイプが最後に変更された時間
4. 必要に応じて、パイプの詳細を表示するか、新しいパイプを作成します。

   - パイプの詳細にアクセスするには:

     パイプ名を選択します。

     これにより、コンソールのパイプの詳細ページが起動します EventBridge。
   - 新しいパイプを作成するには:

     [Amazon SQS キューをパイプに接続] を選択します。

     これにより、Amazon SQS SQS キューをパイプソースとして指定した状態で、コンソールの EventBridgeパイプの作成ページが起動します。詳細については、「Amazon ユーザーガイド EventBridge」の「パイプの作成」を参照してください。 EventBridge

     > **Important**
     >
     > Amazon SQS キュー上のメッセージは、単一のパイプによって読み取られてから、処理された後でキューから削除されます。これは、そのパイプに設定できるフィル

ターにメッセージが一致するかどうかを問わず、実行されます。複数のパイプが同一のキューをソースとして使用するように設定する場合は、十分に注意を払ってください。

拡張クライアントライブラリと Amazon Simple Storage Service を使用した大規模な Amazon SQS メッセージの管理

Amazon SQS Extended Client Library for Java と Amazon SQS Extended Client Library for Python を使用して、大きなメッセージを送信できます。これは、256 KB から最大 2 GB までの大きなメッセージペイロードを消費する場合に特に便利です。どちらのライブラリもメッセージペイロードを Amazon Simple Storage Service バケットに保存し、保存された Amazon S3 オブジェクトのリファレンスを Amazon SQS キューに送信します。

> **Note**
>
> Amazon SQS拡張クライアントライブラリは、スタンダードキューとFIFOキューの両方と互換性があります。

トピック

- Java と Amazon S3 を使用した大規模な Amazon SQS Amazon S3の管理
- Python と Amazon S3 を使用した大規模な Amazon SQS Amazon S3の管理

Java と Amazon S3 を使用した大規模な Amazon SQS Amazon S3の管理

Java および Amazon Simple Storage Service (Amazon S3) 用の Amazon SQS 拡張クライアントライブラリを使用して、大規模な Amazon Simple Queue Service (Amazon SQS) メッセージを管理できます。Amazon S3 これは、256 KB から最大 2 GB までの大きなメッセージペイロードを消費する場合に特に便利です。ライブラリは、メッセージペイロードを Amazon S3 バケットに保存し、保存された Amazon S3 オブジェクトのリファレンスを含むメッセージを Amazon SQS キューに送信します。

Amazon SQS Java用拡張クライアントライブラリを使用して、次のことを実行できます。

- メッセージを常にAmazon S3に保存するか、メッセージのサイズが256KB を超える場合のみ保存するかを指定します。

- S3 バケットに保存されている単一のメッセージオブジェクトを参照するメッセージを送信します。
- Amazon S3 バケットからメッセージオブジェクトを取得する
- Amazon S3 バケットからメッセージオブジェクトを削除する

前提条件

次の例では、 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。

サンプルコードを実行する前に、 AWS 認証情報を設定します。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設定」を参照してください。

SDK for JavaとJava用Amazon SQS 拡張クライアントライブラリには、J2SE Devopment Kit 8.0 以降が必要です。

> **Note**
>
> Java用のAmazon SQS 拡張クライアントライブラリを使用して、のみと AWS SDK for Java のAmazon S3 を使用するAmazon SQSメッセージを管理するために使用することができます。これは AWS CLI、、Amazon SQS コンソール、Amazon SQS HTTP API、またはその他の AWS SDKsでは実行できません。

AWS SDK for Java 2.x の例: Amazon S3 を使用して大きな Amazon SQS メッセージを管理する

次の AWS SDK for Java 2.x の例では、ランダムな名前の Amazon S3 バケットを作成し、14 日後にオブジェクトを完全に削除するライフサイクルルールを追加します。また、名前付きのキューを作成してMyQueueS3 バケットに保存されている256KB を超えるランダムなメッセージをキューに送信します。最後に、コードはメッセージを受信し、そのメッセージに関する情報を返信して、メッセージ、キュー、およびバケットを削除します。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
```

```
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.*;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

public class SQSExtendedClientExample {

// Create an Amazon S3 bucket with a random name.
private final static String S3_BUCKET_NAME = UUID.randomUUID() + "-"
        + DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

public static void main(String[] args) {

    /*
     * Create a new instance of the builder with all defaults (credentials
     * and region) set automatically. For more information, see
     * Creating Service Clients in the AWS SDK for Java Developer Guide.
     */
    final AmazonS3 s3 = AmazonS3ClientBuilder.defaultClient();

    /*
```

```
     * Set the Amazon S3 bucket name, and then set a lifecycle rule on the
     * bucket to permanently delete objects 14 days after each object's
     * creation date.
     */
    final BucketLifecycleConfiguration.Rule expirationRule =
            new BucketLifecycleConfiguration.Rule();
    expirationRule.withExpirationInDays(14).withStatus("Enabled");
    final BucketLifecycleConfiguration lifecycleConfig =
            new BucketLifecycleConfiguration().withRules(expirationRule);

    // Create the bucket and allow message objects to be stored in the bucket.
    s3.createBucket(S3_BUCKET_NAME);
    s3.setBucketLifecycleConfiguration(S3_BUCKET_NAME, lifecycleConfig);
    System.out.println("Bucket created and configured.");

    /*
     * Set the Amazon SQS extended client configuration with large payload
     * support enabled.
     */
    final ExtendedClientConfiguration extendedClientConfig =
            new ExtendedClientConfiguration()
                    .withLargePayloadSupportEnabled(s3, S3_BUCKET_NAME);

    final AmazonSQS sqsExtended =
            new AmazonSQSExtendedClient(AmazonSQSClientBuilder
                    .defaultClient(), extendedClientConfig);

    /*
     * Create a long string of characters for the message object which will
     * be stored in the bucket.
     */
    int stringLength = 300000;
    char[] chars = new char[stringLength];
    Arrays.fill(chars, 'x');
    final String myLongString = new String(chars);

    // Create a message queue for this example.
    final String QueueName = "MyQueue" + UUID.randomUUID().toString();
    final CreateQueueRequest createQueueRequest =
            new CreateQueueRequest(QueueName);
    final String myQueueUrl = sqsExtended
            .createQueue(createQueueRequest).getQueueUrl();
    System.out.println("Queue created.");
```

```
    // Send the message.
    final SendMessageRequest myMessageRequest =
            new SendMessageRequest(myQueueUrl, myLongString);
    sqsExtended.sendMessage(myMessageRequest);
    System.out.println("Sent the message.");

    // Receive the message.
    final ReceiveMessageRequest receiveMessageRequest =
            new ReceiveMessageRequest(myQueueUrl);
    List<Message> messages = sqsExtended
            .receiveMessage(receiveMessageRequest).getMessages();

    // Print information about the message.
    for (Message message : messages) {
        System.out.println("\nMessage received.");
        System.out.println("  ID: " + message.getMessageId());
        System.out.println("  Receipt handle: " + message.getReceiptHandle());
        System.out.println("  Message body (first 5 characters): "
                + message.getBody().substring(0, 5));
    }

    // Delete the message, the queue, and the bucket.
    final String messageReceiptHandle = messages.get(0).getReceiptHandle();
    sqsExtended.deleteMessage(new DeleteMessageRequest(myQueueUrl,
            messageReceiptHandle));
    System.out.println("Deleted the message.");

    sqsExtended.deleteQueue(new DeleteQueueRequest(myQueueUrl));
    System.out.println("Deleted the queue.");

    deleteBucketAndAllContents(s3);
    System.out.println("Deleted the bucket.");
}

private static void deleteBucketAndAllContents(AmazonS3 client) {

    ObjectListing objectListing = client.listObjects(S3_BUCKET_NAME);

    while (true) {
        for (S3ObjectSummary objectSummary : objectListing
                .getObjectSummaries()) {
            client.deleteObject(S3_BUCKET_NAME, objectSummary.getKey());
        }
```

```
        if (objectListing.isTruncated()) {
            objectListing = client.listNextBatchOfObjects(objectListing);
        } else {
            break;
        }
    }

    final VersionListing list = client.listVersions(
            new ListVersionsRequest().withBucketName(S3_BUCKET_NAME));

    for (S3VersionSummary s : list.getVersionSummaries()) {
        client.deleteVersion(S3_BUCKET_NAME, s.getKey(), s.getVersionId());
    }

    client.deleteBucket(S3_BUCKET_NAME);
}
}
```

AWS SDK for Java 2.x の例: Amazon S3 を使用して大きな Amazon SQS メッセージを管理する

次の AWS SDK for Java 2.x の例では、ランダムな名前の Amazon S3 バケットを作成し、14 日後にオブジェクトを完全に削除するライフサイクルルールを追加します。また、名前付きのキューを作成してMyQueueS3 バケットに保存されている256KB を超えるランダムなメッセージをキューに送信します。最後に、コードはメッセージを受信し、そのメッセージに関する情報を返信して、メッセージ、キュー、およびバケットを削除します。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */
```

```
            import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;


/**
 * Examples of using Amazon SQS Extended Client Library for Java 2.x
 *
 */
public class SqsExtendedClientExamples {
    // Create an Amazon S3 bucket with a random name.
    private final static String S3_BUCKET_NAME = UUID.randomUUID() + "-"
            + DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

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

```
        /*
         * Create a new instance of the builder with all defaults (credentials
         * and region) set automatically. For more information, see
         * Creating Service Clients in the AWS SDK for Java Developer Guide.
         */
        final S3Client s3 = S3Client.create();

        /*
         * Set the Amazon S3 bucket name, and then set a lifecycle rule on the
         * bucket to permanently delete objects 14 days after each object's
         * creation date.
         */
        final LifecycleRule lifeCycleRule = LifecycleRule.builder()
                .expiration(LifecycleExpiration.builder().days(14).build())
                .filter(LifecycleRuleFilter.builder().prefix("").build())
                .status(ExpirationStatus.ENABLED)
                .build();
        final BucketLifecycleConfiguration lifecycleConfig =
BucketLifecycleConfiguration.builder()
                .rules(lifeCycleRule)
                .build();

        // Create the bucket and configure it
        s3.createBucket(CreateBucketRequest.builder().bucket(S3_BUCKET_NAME).build());

s3.putBucketLifecycleConfiguration(PutBucketLifecycleConfigurationRequest.builder()
                .bucket(S3_BUCKET_NAME)
                .lifecycleConfiguration(lifecycleConfig)
                .build());
        System.out.println("Bucket created and configured.");

        // Set the Amazon SQS extended client configuration with large payload support
enabled
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration().withPayloadSupportEnabled(s3, S3_BUCKET_NAME);

        final SqsClient sqsExtended = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);

        // Create a long string of characters for the message object
        int stringLength = 300000;
        char[] chars = new char[stringLength];
        Arrays.fill(chars, 'x');
```

```
        final String myLongString = new String(chars);

        // Create a message queue for this example
        final String queueName = "MyQueue-" + UUID.randomUUID();
        final CreateQueueResponse createQueueResponse =
sqsExtended.createQueue(CreateQueueRequest.builder().queueName(queueName).build());
        final String myQueueUrl = createQueueResponse.queueUrl();
        System.out.println("Queue created.");

        // Send the message
        final SendMessageRequest sendMessageRequest = SendMessageRequest.builder()
                .queueUrl(myQueueUrl)
                .messageBody(myLongString)
                .build();
        sqsExtended.sendMessage(sendMessageRequest);
        System.out.println("Sent the message.");

        // Receive the message
        final ReceiveMessageResponse receiveMessageResponse =
sqsExtended.receiveMessage(ReceiveMessageRequest.builder().queueUrl(myQueueUrl).build());
        List<Message> messages = receiveMessageResponse.messages();

        // Print information about the message
        for (Message message : messages) {
            System.out.println("\nMessage received.");
            System.out.println("  ID: " + message.messageId());
            System.out.println("  Receipt handle: " + message.receiptHandle());
            System.out.println("  Message body (first 5 characters): " +
message.body().substring(0, 5));
        }

        // Delete the message, the queue, and the bucket
        final String messageReceiptHandle = messages.get(0).receiptHandle();

sqsExtended.deleteMessage(DeleteMessageRequest.builder().queueUrl(myQueueUrl).receiptHandle(me
        System.out.println("Deleted the message.");


sqsExtended.deleteQueue(DeleteQueueRequest.builder().queueUrl(myQueueUrl).build());
        System.out.println("Deleted the queue.");

        deleteBucketAndAllContents(s3);
        System.out.println("Deleted the bucket.");
```

```
    }

    private static void deleteBucketAndAllContents(S3Client client) {
        ListObjectsV2Response listObjectsResponse =
 client.listObjectsV2(ListObjectsV2Request.builder().bucket(S3_BUCKET_NAME).build());

        listObjectsResponse.contents().forEach(object -> {

 client.deleteObject(DeleteObjectRequest.builder().bucket(S3_BUCKET_NAME).key(object.key()).bui
        });

        ListObjectVersionsResponse listVersionsResponse =
 client.listObjectVersions(ListObjectVersionsRequest.builder().bucket(S3_BUCKET_NAME).build());

        listVersionsResponse.versions().forEach(version -> {

 client.deleteObject(DeleteObjectRequest.builder().bucket(S3_BUCKET_NAME).key(version.key()).ve
        });


 client.deleteBucket(DeleteBucketRequest.builder().bucket(S3_BUCKET_NAME).build());
    }
}
```

Apache Maven を使用して、Java プロジェクト用の Amazon SQS 拡張クライアントを設定および構築したり、SDK 自体を構築したりできます。アプリケーションで使用する SDK から個々のモジュールを指定します。

```
<properties>
    <aws-java-sdk.version>2.20.153</aws-java-sdk.version>
</properties>

<dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>sqs</artifactId>
      <version>${aws-java-sdk.version}</version>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
```

```
      <version>${aws-java-sdk.version}</version>
    </dependency>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>amazon-sqs-java-extended-client-lib</artifactId>
      <version>2.0.4</version>
    </dependency>

    <dependency>
      <groupId>joda-time</groupId>
      <artifactId>joda-time</artifactId>
      <version>2.12.6</version>
    </dependency>
</dependencies>
```

Python と Amazon S3 を使用した大規模な Amazon SQS Amazon S3の管理

Python および Amazon Simple Storage Service 用の Amazon Simple Queue Service 拡張クライアントライブラリを使用して、大きな Amazon SQS メッセージを管理できます。これは、256 KB から最大 2 GB までの大きなメッセージペイロードを消費する場合に特に便利です。ライブラリは、メッセージペイロードを 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含むメッセージを Amazon SQS キューに送信します。

Extended Client Library for Python を使用して、次の操作を実行できます。

- ペイロードを常に Amazon S3 に保存するか、ペイロードサイズが 256 KB を超えた場合にのみ S3 に保存するかを指定します。
- Amazon S3 バケットに保存されている単一のメッセージオブジェクトを参照するメッセージを送信する
- Amazon S3 バケットから対応するペイロードオブジェクトを取得する
- Amazon S3 バケットから対応するペイロードオブジェクトを削除する

前提条件

Python 用 Amazon SQS 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次のとおりです。

- 必要な認証情報を持つ AWS アカウント。 AWS アカウントを作成するには、AWS ホームページ に移動し、 AWS アカウントの作成 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、「 認証情報」を参照してください。
- 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 AWS デベロッパーガイドの SDK for Python ドキュメントを参照してください。
- Python 3.x (またはそれ以降) および `pip`。
- PyPI から入手できる Amazon SQS Extended Client Library for Python

> **Note**
>
> Python 用 Amazon SQS 拡張クライアントライブラリを使用して、 AWS SDK for Python でのみ Amazon S3 を使用して Amazon SQS メッセージを管理できます。 Amazon S3 これは、 AWS CLI、Amazon SQS コンソール、Amazon SQS HTTP API、またはその他の AWS SDKsでは実行できません。

メッセージストレージの設定

Amazon SQS 拡張クライアントは、 が次のメッセージ属性を使用して Amazon S3 メッセージストレージオプションを設定します。

- `large_payload_support`: 大きなメッセージを保存する Amazon S3 バケット名。
- `always_through_s3`: の場合`True`、すべてのメッセージが Amazon S3 に保存されます。の場合`False`、256 KB 未満のメッセージは s3 バケットにシリアル化されません。デフォルトは `False` です。
- `use_legacy_attribute`: の場合`True`、すべての発行済みメッセージは、現在の予約済みメッセージ属性 (`SQSLargePayloadSize`) ではなく、レガシーの予約済みメッセージ属性 () を使用しますExtendedPayloadSize。

Python 用拡張クライアントライブラリを使用した大規模な Amazon SQS メッセージの管理

次の例では、ランダムな名前で Amazon S3 バケットを作成します。次に、 という名前の Amazon SQS キューを作成し、S3 バケットに保存され、256 KB を超えるメッセージをキューMyQueueに送

信します。最後に、コードはメッセージを受信し、そのメッセージに関する情報を返信して、メッセージ、キュー、およびバケットを削除します。

```
import boto3
import sqs_extended_client

#Set the Amazon SQS extended client configuration with large payload.
sqs_extended_client = boto3.client("sqs", region_name="us-east-1")
sqs_extended_client.large_payload_support = "S3_BUCKET_NAME"
sqs_extended_client.use_legacy_attribute = False


# Create an SQS message queue for this example. Then, extract the queue URL.
queue = sqs_extended_client.create_queue(
    QueueName = "MyQueue"
)
queue_url = sqs_extended_client.get_queue_url(
    QueueName = "MyQueue"
)['QueueUrl']


# Create the S3 bucket and allow message objects to be stored in the bucket.
sqs_extended_client.s3_client.create_bucket(Bucket=sqs_extended_client.large_payload_support)

# Sending a large message
small_message = "s"
large_message = small_message * 300000 # Shall cross the limit of 256 KB

send_message_response = sqs_extended_client.send_message(
    QueueUrl=queue_url,
    MessageBody=large_message
)
assert send_message_response['ResponseMetadata']['HTTPStatusCode'] == 200

# Receiving the large message
receive_message_response = sqs_extended_client.receive_message(
    QueueUrl=queue_url,
    MessageAttributeNames=['All']
)
assert receive_message_response['Messages'][0]['Body'] == large_message
receipt_handle = receive_message_response['Messages'][0]['ReceiptHandle']
```

```
# Deleting the large message
# Set to True for deleting the payload from S3
sqs_extended_client.delete_payload_from_s3 = True
delete_message_response = sqs_extended_client.delete_message(
    QueueUrl=queue_url,
    ReceiptHandle=receipt_handle
)

assert delete_message_response['ResponseMetadata']['HTTPStatusCode'] == 200

# Deleting the queue
delete_queue_response = sqs_extended_client.delete_queue(
    QueueUrl=queue_url
)

assert delete_queue_response['ResponseMetadata']['HTTPStatusCode'] == 200
```

Amazon SQS コンソールを使用した Amazon SQS キューの設定

Amazon SQS コンソールを使用して、Amazon Simple Queue Service (Amazon SQS) キューと機能を設定し、管理します。コンソールを使用して、サーバー側の暗号化、デッドレターキューとキューの関連付け、 AWS Lambda 関数を呼び出すトリガーの設定などの機能を設定することもできます。

トピック

Amazon SQS の属性ベースのアクセスコントロール

ABAC とは

属性ベースのアクセスコントロール (ABAC) は、ユーザーと AWS リソースにアタッチされているタグに基づいてアクセス許可を定義する認証プロセスです。ABAC は、属性と値に基づいてきめ細かく柔軟なアクセス制御を実現し、再構成されたロールベースのポリシーに関連するセキュリティリスクを軽減し、監査とアクセスポリシー管理を一元化します。ABAC の詳細については、「IAM ユーザーガイド」の「AWSの ABAC とは」を参照してください。

Amazon SQS は、Amazon SQS キューに関連付けられているタグとエイリアスに基づいて Amazon SQS キューへのアクセスを制御できるようにすることで、ABAC をサポートしています。Amazon

SQS の ABAC を有効にするタグおよびエイリアスの条件キーは、ポリシーを編集したりグラントを管理することなく、IAM プリンシパルが Amazon SQS キューを使用することを許可します。

ABAC では、タグを使用して Amazon SQS キューの IAM アクセス許可とポリシーを設定できます。これにより、アクセス許可管理をスケールできます。各ビジネスロールに追加するタグを使用して、IAM で単一のアクセス許可ポリシーを作成できます。新しいリソースを追加するたびにポリシーを更新する必要はありません。IAM プリンシパルにタグをアタッチして ABAC ポリシーを作成することもできます。呼び出しを行う IAM ユーザーロールのタグが Amazon SQS キュータグと一致した場合に Amazon SQS オペレーションを許可するように ABAC ポリシーを設計できます。でのタグ付けの詳細については AWS、AWS 「タグ付け戦略」および「」を参照してくださいAmazon SQSコスト配分タグ。

> **Note**
>
> Amazon SQS の ABAC は現在、Amazon SQS が利用可能なすべての AWS 商用リージョンで利用できますが、以下の例外があります。
>
> - 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)
> - 欧州 (スペイン)
> - 欧州 (チューリッヒ)

Amazon SQS で ABAC を使用すべき理由は何ですか。

Amazon SQS で ABAC を使用すると、以下のようなメリットがあります。

- Amazon SQS 用 ABAC では、必要なアクセス許可ポリシーが少なくて済みます。職務機能ごとに異なるポリシーを作成する必要はありません。複数のキューに適用されるリソースタグとリクエストタグを使用できるため、運用上のオーバーヘッドが軽減できます。
- ABAC を使用すると、チームを迅速にスケールできます。リソースの作成時に適切なタグが付けられると、新しいリソースのアクセス許可はタグに基づいて自動的に付与されます。
- IAM プリンシパルのアクセス許可を使用して、リソースへのアクセスを制限します。IAM プリンシパルのタグを作成し、そのタグを使用して IAM プリンシパルのタグと一致する特定のアクションへのアクセスを制限できます。これにより、リクエストのアクセス許可を付与するプロセスを自動化できます。

- リソースにアクセスしているユーザーを追跡できます。セッションの ID は、 AWS CloudTrailのユーザー属性を調べることで判断できます。

トピック

- Amazon SQS の ABAC 条件キー
- Amazon SQS でのアクセスコントロールのタグ付け
- IAM ユーザーと Amazon SQS キューの作成
- 属性ベースのアクセス制御のテスト

Amazon SQS の ABAC 条件キー

関数のアクションは、以下の条件キーを使用して制御できます。

ABAC 条件キー	説明	ポリシータイプ	Amazon SQS オペレーション
aws:ResourceTag	Amazon SQS キューのタグ (キーと値) が、ポリシーのタグ (キーと値) またはタグパターンと一致する	IAM ポリシーのみ	Amazon SQS キューリソースのオペレーション
aws:RequestTag	Amazon SQS キューリソースのオペレーションのタグ (キーと値) が、ポリシーのタグ (キーと値) またはタグパターンと一致する	キューポリシーと IAM ポリシー	TagQueue, UntagQueue, CreateQueue
aws:TagKeys	リクエスト内のタグキーが、ポリシーのタグキーと一致する	キューポリシーと IAM ポリシー	TagQueue, UntagQueue, CreateQueue

Amazon SQS でのアクセスコントロールのタグ付け

タグをアクセス制御に使用する方法の例を以下に示します。IAM ポリシーは、キー environment および値 production のタグを持つリソースタグを含むすべてのキューに対して、IAM ユーザーにすべての Amazon SQS アクションを制限します。詳細については、「タグを使用した属性ベースのアクセスコントロール」および AWS 「組織」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessForProd",
      "Effect": "Deny",
      "Action": "sqs:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/environment": "prod"
        }
      }
    }
  ]
}
```

IAM ユーザーと Amazon SQS キューの作成

次の例では、 AWS Management Console と を使用して Amazon SQS へのアクセスを制御する ABAC ポリシーを作成する方法を説明します AWS CloudFormation。

の使用 AWS Management Console

IAM ユーザーの作成

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. 左側のナビゲーションペインで [ユーザー] を選択します。
3. [ユーザーを追加] を選択し、[ユーザー名] テキストボックスに名前を入力します。
4. [アクセスキー - プログラムによるアクセス] ボックスを選択し、[次へ: 許可] を選びます。
5. [Next:Tags] (次のステップ: タグ) を選択します。

6. タグのキーを environment、タグの値を beta として追加します。
7. [次の手順: 確認]、[作成] の順に選択します。
8. アクセスキー ID とシークレットアクセスキーを安全な場所にコピーして保存します。

IAM ユーザーのアクセス許可を追加する

1. 作成した IAM ユーザーを選択します。
2. [Add inline policy] (インラインポリシーの追加) を選択します。
3. [JSON] タブに、以下のポリシーを貼り付けます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessForSameResTag",
      "Effect": "Allow",
      "Action": [
        "sqs:SendMessage",
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/environment": "${aws:PrincipalTag/environment}"
        }
      }
    },
    {
      "Sid": "AllowAccessForSameReqTag",
      "Effect": "Allow",
      "Action": [
        "sqs:CreateQueue",
        "sqs:DeleteQueue",
        "sqs:SetQueueAttributes",
        "sqs:tagqueue"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/environment": "${aws:PrincipalTag/environment}"
```

```
        }
      }
    },
    {
      "Sid": "DenyAccessForProd",
      "Effect": "Deny",
      "Action": "sqs:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/stage": "prod"
        }
      }
    }
  ]
}
```

4. [ポリシーの確認] を選択します。
5. [ポリシーの作成] を選択します。

の使用 AWS CloudFormation

次のサンプル AWS CloudFormation テンプレートを使用して、インラインポリシーと Amazon SQS キューがアタッチされた IAM ユーザーを作成します。

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "CloudFormation template to create IAM user with custom inline policy"
Resources:
    IAMPolicy:
        Type: "AWS::IAM::Policy"
        Properties:
            PolicyDocument: |
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Sid": "AllowAccessForSameResTag",
                            "Effect": "Allow",
                            "Action": [
                                "sqs:SendMessage",
                                "sqs:ReceiveMessage",
                                "sqs:DeleteMessage"
```

```
                            ],
                            "Resource": "*",
                            "Condition": {
                                "StringEquals": {
                                    "aws:ResourceTag/environment": "${aws:PrincipalTag/
environment}"
                                }
                            }
                        },
                        {
                            "Sid": "AllowAccessForSameReqTag",
                            "Effect": "Allow",
                            "Action": [
                                "sqs:CreateQueue",
                                "sqs:DeleteQueue",
                                "sqs:SetQueueAttributes",
                                "sqs:tagqueue"
                            ],
                            "Resource": "*",
                            "Condition": {
                                "StringEquals": {
                                    "aws:RequestTag/environment": "${aws:PrincipalTag/
environment}"
                                }
                            }
                        },
                        {
                            "Sid": "DenyAccessForProd",
                            "Effect": "Deny",
                            "Action": "sqs:*",
                            "Resource": "*",
                            "Condition": {
                                "StringEquals": {
                                    "aws:ResourceTag/stage": "prod"
                                }
                            }
                        }
                    ]
                }

            Users:
              - "testUser"
            PolicyName: tagQueuePolicy
```

```
    IAMUser:
        Type: "AWS::IAM::User"
        Properties:
            Path: "/"
            UserName: "testUser"
            Tags:
              -
                Key: "environment"
                Value: "beta"
```

属性ベースのアクセス制御のテスト

以下の例は、Amazon SQS で属性ベースのアクセス制御をテストする方法について示しています。

タグキーを environment に設定し、タグ値を prod に設定してキューを作成します。

この AWS CLI コマンドを実行して、タグキーを environment に設定し、タグ値を prod に設定してキューの作成をテストします。 AWS CLI がない場合は、マシン用にダウンロードして設定できます。

```
aws sqs create-queue --queue-name prodQueue –region us-east-1 –tags "environment=prod"
```

Amazon SQS エンドポイントから AccessDenied エラーを受け取ります。

```
An error occurred (AccessDenied) when calling the CreateQueue operation: Access to the
 resource <queueUrl> is denied.
```

これは、IAM ユーザーのタグ値が CreateQueue API コールで渡されたタグと一致しないためです。キーを environment に、値を beta に設定したタグを IAM ユーザーに適用したことを思い出してください。

タグキーを environment に設定し、タグ値を beta に設定してキューを作成する

この CLI コマンドを実行して、タグキーを environment に設定し、タグ値を beta に設定したキューの作成をテストします。

```
aws sqs create-queue --queue-name betaQueue –region us-east-1 –tags "environment=beta"
```

キューが正常に作成されたことを確認する次のようなメッセージが表示されます。

```
{
"QueueUrl": "<queueUrl>“
}
```

キューへメッセージを送信する

この CLI コマンドを実行して、キューへのメッセージ送信をテストします。

```
aws sqs send-message --queue-url <queueUrl> --message-body testMessage
```

レスポンスには、Amazon SQS キューへのメッセージ配信が成功したことが示されます。IAM ユーザーアクセス許可により、beta タグを持つキューにメッセージを送信できます。レスポンスにはメッセージを含む MD5OfMessageBody と MessageId が含まれます。

```
{
"MD5OfMessageBody": "<MD5OfMessageBody>",
"MessageId": "<MessageId>"
}
```

Amazon SQS コンソールを使用したキューパラメータの設定

キューを作成または編集する時、以下のパラメータを設定できます。

- 可視性タイムアウト— キューから受信したメッセージが (あるコンシューマによって) 他のメッセージコンシューマーに表示されない時間の長さ。詳細については、「可視性」を参照してください。

  > **Note**
  >
  > コンソールを使用して可視性タイムアウトを設定すると、キュー内のすべてのメッセージのタイムアウト値が設定されます。単一または複数のメッセージのタイムアウトを設定するには、いずれかの AWS SDKsを使用する必要があります。

- メッセージの保持期間— キューに残っているメッセージを Amazon SQS が保持する時間は。デフォルトでは、キューは4日間メッセージを保持します。最大14日間までメッセージを保持するようにキューを設定できます。詳細については、「メッセージの保持期間」を参照してください。
- 配信の遅延—キューに追加されたメッセージを配信する前に Amazon SQSが遅延する時間。詳細については、「配信の遅延」を参照してください。

- メッセージの最大サイズ—このキューの最大メッセージサイズ。詳細については、「メッセージの最大サイズ」を参照してください。
- メッセージの受信待ち時間—キューが受信リクエストを受け取った後、Amazon SQS がメッセージが使用可能になるまでの待機する最大時間。詳細については、「Amazon SQSショートポーリングとロングポーリング」を参照してください。
- コンテンツベースの重複除外を有効—Amazon SQS は、メッセージの本文に基づいて重複除外IDを自動的に作成できます。詳細については、「Amazon SQS での FIFO キューの開始方法」を参照してください。
- 高スループット FIFOを有効にする—キューのメッセージの高スループットを有効にするために使用します。このオプションを選択すると、関連するオプションが (重複除外スコープそしてFIFOスループットの制限) を使用して、FIFOキューの高スループットを有効にするために必要な設定に変更されます。詳細については、「Amazon SQS の FIFO キューの高スループット」および「Amazon SQS メッセージクォータ」を参照してください。
- ポリシーのリドライブ許可:どのソースキューがこのキューをデッドレターキューとして使用できるを定義します。詳細については、「Amazon SQS でデッドレターキューを使用します 」を参照してください。

既存のキュー (コンソール)のキューパラメータを設定するには

1. Amazon SQS コンソール を開きますhttps://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)]を選択します。キューを選択し、編集を選択します。
3. 設定セクションにスクロールします。
4. 可視性タイムアウトを使用する場合、期間と単位を入力します。範囲は0秒から12時間です。デフォルト値は30秒です。
5. メッセージの保持期間を使用する場合、期間と単位を入力します。有効範囲は1分から14日です。デフォルト値は4日です。
6. スタンダードキューの場合は、メッセージの受信待ち時間の値を入力します。範囲は 0から20秒です。デフォルト値は0秒で、ショートポーリングで設定します。0以外の値を指定すると、ロングポーリングが設定されます。
7. 配信の遅延を使用する場合、期間と単位を入力します。範囲は0秒から15 分です。デフォルト値は0 秒です。
8. メッセージの最大サイズ を使用する場合、値を入力します。範囲は1KBから256KBです。デフォルト値は 256 KBです。

9. FIFO キューの場合は、コンテンツベースの重複除外を有効にするためにコンテンツベースの重複除外を有効にします。デフォルト設定は無効です。
10. (オプション) FIFO キューの場合、キュー内のメッセージの送受信でより高スループットを有効にするには、[高スループット FIFO を有効にする] を選択します。

    このオプションを選択すると、関連するオプション (重複除外のスコープおよびFIFO スループットの制限) を使用して、FIFO キューの高スループットを有効にするために必要な設定に変更されます。高スループット FIFO の使用に必要な設定のいずれかを変更すると、キューに対して通常のスループットが有効になり、指定されたとおりに重複除外が実行されます。詳細については、「Amazon SQS の FIFO キューの高スループット」および「Amazon SQS メッセージクォータ」を参照してください。
11. リドライブ許可ポリシーでは、[有効] を選択します。[すべて許可] (デフォルト)、[キュー別]、または [すべて拒否] から選択します。キュー別を選んだとき、 Amazon リソースネーム (ARN) で最大10個のソースキューのリストを指定します。
12. キューパラメータの設定が完了したら、保存を選択します。

アクセスポリシーの設定

キューの編集時、そのアクセスポリシーを設定できます。

アクセスポリシーは、キューにアクセスできるアカウント、ユーザー、および役割を定義します。アクセスポリシーでは、ユーザーがアクセスできるアクションも定義します (`SendMessage`、`ReceiveMessage`、 または`DeleteMessage`など) 。デフォルトポリシーでは、キューの所有者のみがメッセージの送受信を許可します。

既存のキュー(コンソール)のアクセスポリシーを設定

1. Amazon SQS コンソールを開きますhttps://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)]を選択します。
3. キューを選択し、編集を選びます。
4. アクセスポリシーセクションにスクロール。
5. 入力ボックスのアクセスポリシーステートメントを編集します。アクセスポリシーステートメントの詳細については、「」を参照してくださいAmazon SQSでの Identity and Access Management。
6. アクセスポリシーの設定が完了したら、保存を選びます。

SQS マネージド暗号化キーを使用したキューのサーバー側の暗号化の設定

Amazon SQS マネージド SSE (SSE-SQS) では、Amazon SQS が管理するデフォルトのサーバー側の暗号化オプションに加えて、SQS マネージド暗号化キーを使用して、メッセージキューを介して送信される機密データを保護する、カスタムマネージドサーバー側の暗号化を作成できます。SSE-SQSでは、暗号化キーの作成および管理、データを暗号化のためにコードを修正する必要はありません。SSE-SQSを使用すると、データを安全に送信し、追加の費用なしで厳格な暗号化コンプライアンスと規制要件に対応できます。

SSE-SQS は、256 ビットの高度暗号化スタンダード (AES-256)の 暗号化を使用して保存中のデータを保護します。SSEはAmazon SQSサービスがメッセージを受信するとすぐに暗号化します。Amazon SQS は暗号化された形式でメッセージを保存し、承認済みのコンシューマーに送信した場合のみ解読されます。

> **Note**
>
> - デフォルトの SSE オプションは、暗号化属性を指定せずにキューを作成した場合にのみ有効です。
> - Amazon SQS では、キューの暗号化をすべてオフにすることができます。したがって、KMS-SSE をオフにしても SQS-SSE は自動的には有効になりません。KMS-SSE を無効にした後に SQS-SSE を有効にする場合は、リクエストに属性の変更を追加する必要があります。

キューにSSE-SQS暗号化を設定するには(コンソール)

> **Note**
>
> HTTP (非 TLS) エンドポイントを使用して作成された新しいキューは、デフォルトで SSE-SQS 暗号化を有効化しません。HTTPS または署名バージョン 4 エンドポイントを使用して Amazon SQS キューを作成するのがセキュリティのベストプラクティスです。

1. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)] を選択します。
3. キューを選択し、{編集}を選択します。

4. 暗号化を拡張します。
5. サーバー側の暗号化を使用するには、[有効] (デフォルト) を選択します。

> **Note**
>
> SSE を有効にすると、暗号化されたキューへの匿名の SendMessage リクエストと ReceiveMessage リクエストは拒否されます。Amazon SQS のセキュリティベストプラクティスでは、匿名リクエストを使用しないことを推奨しています。Amazon SQS キューに匿名リクエストを送信する場合は、必ず SSE を無効にしてください。

6. Amazon SQSキー (SSE-SQS)を選択します。このオプションは追加料金なしで使用されます。
7. [保存] を選択します。

Amazon SQS コンソールを使用したキューのサーバー側の暗号化の設定

キューのメッセージのデータを保護するために、Amazon SQS では、新しく作成されたキューのサーバー側の暗号化がデフォルトで有効になっています。Amazon SQS は、Amazon Web Services キー管理サービス (Amazon Web Services KMS) と統合してサーバー側の暗号化 (SSE) 用の KMS キーを管理します。SSE の使用詳細については、「Amazon SQS での保管時の暗号化」をご参照ください。

キューに割り当てる KMS キーは、キューの使用が承認されたすべてのプリンシパルの許可を含むキーポリシーが必要です。詳細については、キー管理をご参照ください。

KMS 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 SQS コンソールで KMS キーに関する情報を閲覧できません。これらの許可を付与するように、KMS キーの所有者へ依頼してください。詳しい情報については、キー管理をご参照ください。

キューの作成または編集する場合は、SSE-KMS を設定できます。

既存のキューに SSE-KMSを設定するには(コンソール)

1. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)] を選択します。
3. キューを選択し、{編集}を選択します。

4. 暗号化を拡張します。
5. サーバー側の暗号化を使用するには、[有効] (デフォルト) を選択します。

> **Note**
>
> SSE を有効にすると、暗号化されたキューへの匿名の SendMessage リクエストと ReceiveMessage リクエストは拒否されます。Amazon SQS のセキュリティベストプラクティスでは、匿名リクエストを使用しないことを推奨しています。Amazon SQS キューに匿名リクエストを送信する場合は、必ず SSE を無効にしてください。

6. AWS キーマネジメントサービスキー (SSE-KMS)を選びます。

   コンソールは [Description] (概要)、[Account] (アカウント)、KMS キーの [KMS key ARN] (KMS キー ARN) を表示します。
7. キューの KMS キー ID 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。

   a. [Choose a KMS key alias] (KMS キーのエイリアスを選択) オプションを選択します。
   b. デフォルトキーは、Amazon SQS 用の Amazon Web Services マネージド KMS キーです。このキーを使用する場合、[KMS key] (KMS キー) リストから選択してください。
   c. Amazon Web Services アカウントでカスタム KMS キーを使用する場合、[KMS key] (KMS キー) リストから選択してください。カスタム KMS キーの作成手順については、Amazon Web Services キー管理サービスデベロッパーガイドのキー作成をご参照ください。
   d. リストにないカスタム KMS キーを使用、または別の Amazon Web Services アカウントのカスタム KMS キーを使用する場合、[Enter the KMS key alias] (KMS キーのエイリアス入力) を選択し、KMS キー Amazon リソースネーム (ARN) を入力します。
8. (任意) [Data key reuse period] (データキー再利用期間) について、1 分から 24 時間の範囲内で値を指定します。デフォルトは 5 分です。詳細については、「データキー再利用期間について」を参照してください。
9. SSE-KMS の設定が完了したら、保存を選択します。

Amazon SQS コンソールを使用したキューのコスト配分タグの設定

Amazon SQSキューを整理および識別しやすくするために、コスト配分タグを追加することができます。詳細については、「Amazon SQSコスト配分タグ」を参照してください。

キューの詳細ページでは、タグ付けタブには、キューのタグが表示されます。

キューを作成または編集する場合は、そのタグを設定できます。

既存のキューのタグを設定するには(コンソール)

1. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)]を選択します。
3. キューを選択し、編集を選択します。
4. タグセクションまでスクロールします。
5. キュータグを追加、変更、または削除するには
   a. タグを追加するには、{新しタグを追加}を選択して{キーを}と{値}を入力し、{新しいタグの追加}を選択します。
   b. タグを更新するには、変更します。キーと値を変更します。
   c. タグを削除するには、キー値ペアの横にある {削除} を選択します。
6. タグの設定が完了したら、保存を選択します。

Amazon SQS コンソールを使用した Amazon SNS トピックへのキューのサブスクライブ Amazon SQS

1 つ以上の Amazon SQS キューを Amazon Simple Notification Service (Amazon SNS) トピックにサブスクライブできます。メッセージがトピックに発行されると、Amazon SNSはサブスクライブされたすべてのキューにメッセージを送信します。Amazon SQSは、サブスクリプションと必要なアクセス許可を管理します。Amazon SNS の詳細については、Amazon Simple Notification Service デベロッパーガイドの、「Amazon SNS とは」を参照してください。

Amazon SQSキューをSNSトピックにサブスクライブすると、Amazon SNSはHTTPSを使用してAmazon SQSにメッセージを転送します。暗号化された Amazon SQSキューでの Amazon SNS の使用については、「AWS サービスの KMS アクセス許可を設定する」を参照してください。

> **Important**
>
> Amazon SQS は、アクセスポリシーごとに最大 20 個のステートメントをサポートします。Amazon SNS トピックをサブスクライブすると、そのようなステートメントが 1 つ追加されます。この量を超えると、トピックサブスクリプション配信が失敗します。

キューをSNSにトピックにサブスクライブするには (コンソール)

1. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/ 。
2. ナビゲーションペインで [Queues(キュー)] を選択します。
3. キューの一覧から、SNSトピックをサブスクライブするキューを選択します。
4. {アクション} メニューから、[Amazon SNSトピックにサブスクライブする] を選択します。
5. このキューで使用できる Amazon SNSトピックを特定するメニューから、キューのSNSトピックを選択します。

   SNSトピックがメニューに表示されていない場合は、Amazon SNSトピックARNと入力し、次に、トピックのAmazon リソースネーム (ARN) を入力します。
6. [保存] を選択します。
7. サブスクリプションの結果を検証するには、トピックに発行し、トピックからキューに送信されたメッセージを確認できます。詳細については、Amazon SNS メッセージ発行」を参照してください。

Amazon SQS キューと SNS トピックが異なる にある場合 AWS アカウント、トピック所有者はまずサブスクリプションを確認する必要があります。詳細については、Amazon Simple Notification Service デベロッパーガイドの「Amazon SNSトピックへのサブスクライブ」を参照してください。

クロスリージョン SNS トピックへのサブスクライブの詳細については、Amazon Simple Notification Service デベロッパーガイドの「別のリージョンの Amazon SQS キューまたは関数への Amazon SNS Amazon SQS メッセージ AWS Lambdaの送信」を参照してください。

AWS Lambda 関数をトリガーするための Amazon SQS キューの設定

AWS Lambda 関数を使用して、Amazon SQS キュー内のメッセージを処理できます。Lambda はキューをポーリングし、Lambda関数を、キューメッセージを含むイベントと共に同期的に呼び出します。関数がレコードの各バッチを処理するために十分な時間を取るため、ソースキューの可視性タイムアウトは、関数に設定したタイムアウトの少なくとも 6 倍にに設定してください。追加の時間は、関数が前のバッチの処理中にスロットリングされた場合に、Lambda が再試行することを可能にします。

Lambda関数が処理できないメッセージのデッドレターキューとして動作する別のキューを指定することができます。

Lambda関数は、複数のキューから項目を処理できます(各キューに対して1つのLambda イベントソースを使用)。複数のLambda関数で同じキューを使用できます。

暗号化されたキューを Lambda関数に関連付けても Lambda がメッセージをポーリングしない場合は、`kms:DecryptLambda` 実行ロールへのアクセスを許可します。

以下の制限があることに注意してください:

- キューと Lambda 関数は同じ AWS リージョンに存在する必要があります。
- デフォルトキー (Amazon SQS のAWS マネージド KMS キー) を使用する暗号化されたキューは、別の で Lambda 関数を呼び出すことはできません AWS アカウント。 Amazon SQS

Lambda 関数の実装の詳細については、「 AWS Lambda デベロッパーガイド」のAmazon SQS AWS Lambda での の使用」を参照してください。

前提条件

Lambda関数トリガーを設定するには、以下の要件を満たしている必要があります。

- ユーザーを使用する場合、Amazon SQSロールに以下のアクセス許可が含まれている必要があります。
  - `lambda:CreateEventSourceMapping`
  - `lambda:ListEventSourceMappings`
  - `lambda:ListFunctions`
- Lambda実行ロールに以下のアクセス許可が含まれている必要があります。
  - `sqs:DeleteMessage`
  - `sqs:GetQueueAttributes`
  - `sqs:ReceiveMessage`
- 暗号化されたキューを Lambda 関数に関連付ける場合は、`kms:DecryptLambda` 実行ロールへのアクセス許可をする必要があります。

詳細については、「Amazon SQS でのアクセス管理の概要」を参照してください。

Lambda関数(コンソール)をトリガーするためにキューを設定

1. Amazon SQSコンソール を開きますhttps://console.aws.amazon.com/sqs/。
2. ナビゲーションペインで [Queues(キュー)] を選択します。

3. [キュー]ページで、設定するキューを選択します。
4. キューのページでLambdaトリガータブを選択します。
5. Lambdaトリガーページで、Lambda トリガーを選択します。

   リストに、必要な Lambdaトリガーが含まれていない場合は、Lambda関数トリガーを設定するを選択します。Lambda関数のAmazon リソースネーム (ARN)を入力するか、既存のリソースを選択します。次に、[Save]を選択します。
6. [保存] を選択します。コンソールは設定を保存し、キューのの詳細ページを表示します。

   詳細ページでLambda トリガータブに Lambda 関数とそのステータスが表示されます。Lambda関数がキューに関連付けられるまで約1分かかります。
7. 設定の結果を確認するには、キューにメッセージを送信して、トリガーされたLambdaコンソールの Lambda関数が表示できます。

Amazon を使用して AWS のサービスから Amazon SQS への通知を自動化する EventBridge

Amazon EventBridge では、 AWS サービスを自動化し、アプリケーションの可用性の問題やリソースの変更などのシステムイベントに対応できます。 AWS サービスからのイベントは、 EventBridge ほぼリアルタイムで に配信されます。どのイベントに興味があるのか、イベントがルールに一致した場合にどのように自動的に実行するアクションをとるのか簡単なルールを指定して書き込みすることができます。

EventBridge では、JSON 形式でイベントを受信する Amazon SQS 標準キューや FIFO キューなど、さまざまなターゲットを設定できます。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。

属性を含むメッセージの送信

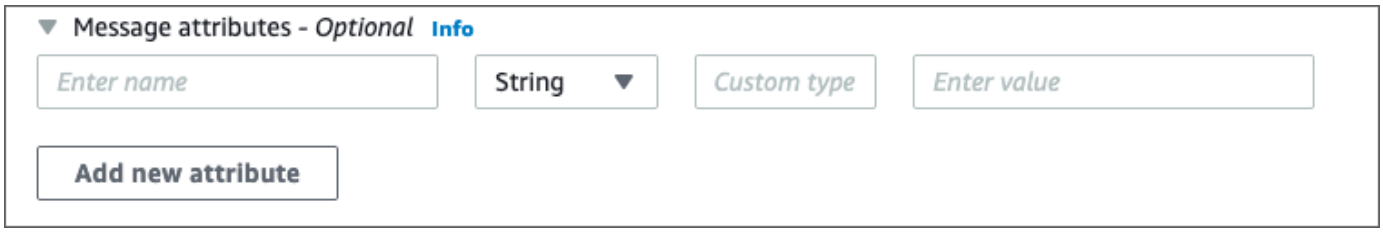
スタンダードとFIFOキューの場合、構造化メタデータ(タイムスタンプ、地理空間データ、署名、識別子など) をメッセージに含めることができます。詳細については、「Amazon SQSメッセージ属性」を参照してください。

Amazon SQS コンソールを使用して属性を含むメッセージをキューに送信するには

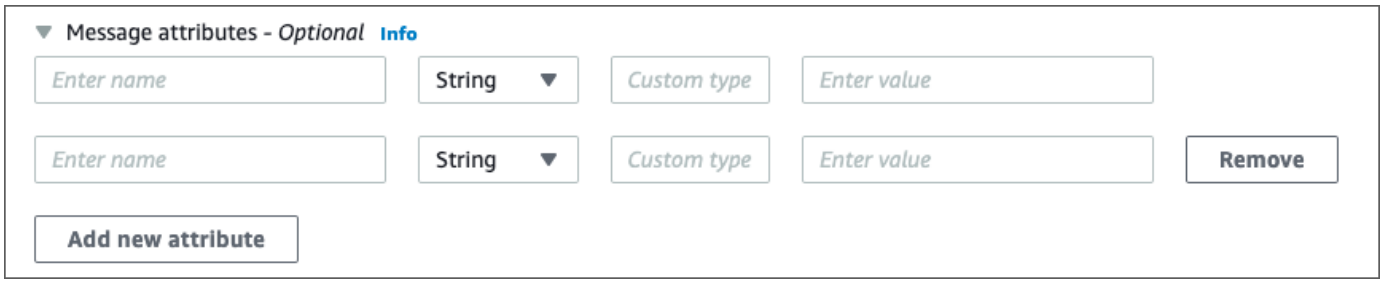
1. Amazon SQSコンソールを開きます https://console.aws.amazon.com/sqs/。

2. ナビゲーションペインで [Queues(キュー)]を選択します。
3. キュー ページで、キュー を選択します。
4. [メッセージの送信と受信] を選択します。
5. メッセージ属性のパラメータを入力します。

   a. {名前テキストボックス}に、[固有の名前]を256文字以下で(入力)します。
   b. 属性タイプについては、文字列、数値、 またはバイナリを選択します。
   c. (オプション) カスタムデータタイプを入力します。たとえば、**byte**、**int**、 または**float**のカスタムデータタイプとして数値を追加することができます。
   d. {値]}テキストボックスに、[メッセージ属性の値]を(入力)します。



6. 属性を追加するには、{属性の追加} を(選択)します。



7. メッセージを送信する前に属性の値をいつでも変更することができます。
8. 属性を削除するには、{削除}を(選択)します。最初の属性を削除するには、{メッセージ属性}を(選択)します。
9. メッセージへの属性の追加を終了したら、{メッセージの送信} を(選択)します。メッセージが送信され、コンソールに成功メッセージが表示されます。送信したメッセージのメッセージ属性に関する情報を表示するには、{詳細を表示}を選択します。{実行}を (選択) してメッセージの詳細ダイアログボックス閉じます。

Amazon SQS のベストプラクティス

これらのベストプラクティスは Amazon SQSを最大限に活用するために役立ちます。

トピック

Amazon SQS 標準および FIFOキューに関する推奨事項

次のベストプラクティスは、Amazon SQS を使用してコストを削減し、効率的にメッセージを処理するのに役立ちます。

トピック

Amazon SQSメッセージの操作

次のガイドラインは、Amazon SQSを使用して効率的にメッセージを処理するのに役立ちます。

トピック

タイムリーな方法でのメッセージの処理

可視性タイムアウトの設定は、アプリケーションがメッセージを処理し、削除するのにかかる時間によって異なります。たとえば、アプリケーションでメッセージを処理するには 10 秒必要だが、可視性タイムアウトを 15 分に設定した場合、前のメッセージ処理に失敗すると、再度処理されるまでに長時間待つ必要があります。または、アプリケーションでメッセージを処理するには 10 秒必要だが、可視性タイムアウトを 2 秒に設定した場合は、元のコンシューマーがメッセージを処理している間に別のコンシューマーより重複メッセージが送信されます。

メッセージの処理に十分な時間を確保するために、次のいずれかの方法を使用します:

- メッセージの処理にかかる時間がわかっている (または合理的に見積もることができる) 場合は、メッセージの可視性タイムアウトを、メッセージの処理と削除にかかる最大時間に拡張します。詳細については、{可視性タイムアウトの構成 } を参照してください。
- メッセージの処理にかかる時間がわからない場合は、コンシューマプロセスについてハートビートを作成:初期可視性タイムアウト (2 分など) を特定し、コンシューマがメッセージで作業している限り、可視性タイムアウトを 1 分ごとに 2 分延長します。

> **Important**
>
> 最大可視性タイムアウトは、ReceiveMessageAmazon SQS がメッセージを受信してから 12 時間です。可視性タイムアウトを延長しても、最大 12 時間はリセットされません。さらに、ReceiveMessageリクエストがタイマーを開始してから 12 時間 (43,200 秒など) は、個々のメッセージのタイムアウトを設定できない場合があります。例えば、メッセージを受信し、43,200 秒VisibilityTimeoutに等しいのChangeMessageVisibility呼び出しを送信して最大 12 時間をすぐに設定すると、失敗する可能性があります。ただし、経由でメッセージをリクエストReceiveMessageしてから可視性タイムアウトを更新するまでに大幅な遅延がない限り、43,195 秒の値を使用すると機能します。コンシューマーが 12 時間以上必要とする場合は、Step Functionsの使用を検討してください。

リクエストエラーの処理

リクエストのエラーを処理するには、次の方法のいずれかを使用します:

- AWS SDK を使用する場合、自動再試行とバックオフのロジックが既に用意されています。詳細については、「Amazon Web Services 全般のリファレンス」の「エラーの再試行と AWSでのエクスポネンシャルバックオフ」を参照してください。
- 再試行とバックオフに AWS SDK 機能を使用しない場合は、Amazon SQS からメッセージ、タイムアウト、またはエラーメッセージを受信せずにReceiveMessageアクションを再試行する前に一時停止 (200 ミリ秒など) してください。 Amazon SQS 同じ結果が得られる ReceiveMessage をそれ以降に使用するには、それよりも長い一時停止 (たとえば、400ms) を許可します。

ロングポーリングのセットアップ

待ち時間がいつになるかReceiveMessageAPI アクションが0より大きい場合ロングポーリングが有効です。長いポーリングの最大待機時間は20秒です。ロングポーリングは、空のレスポンス (ReceiveMessagerequestに対して利用可能なメッセージが含まれていない場合) や偽のレスポンス (利用可能であってもレスポンスに含まれない場合) の数を減らすことにより Amazon SQS を使用するコストを削減することに役立ちます。詳細については、「Amazon SQSショートポーリングとロングポーリング」を参照してください。

最適なメッセージ処理を行うには、次の方法を使用します。

- ほとんどの場合、ReceiveMessage 待機時間を 20 秒に設定します。アプリケーションの設定時間として 20 秒が長すぎる場合は、ReceiveMessage 待機時間の値を小さくします (最小 1 秒)。AWS SDK を使用して Amazon SQS にアクセスしない場合、または AWS SDK の待機時間が短いように設定する場合は、長いリクエストを許可するか、長いポーリングに短い待機時間を使用するように Amazon SQS クライアントを変更する必要がある場合があります。
- 複数のキューにロングポーリングを実行する場合は、すべてのキューに単一スレッドを使用せずに、キューごとに 1 つのスレッドを使用します。キューごとに 1 つのスレッドを使用した場合は、メッセージが使用可能になると、アプリケーションはキューごとにメッセージを処理できるのに対し、複数のキューを単一スレッドでポーリングすると、使用可能なメッセージがないキューを待機 (最大 20 秒) している間、アプリケーションは他のキューで使用可能なメッセージを処理できません。

> **Important**
>
> HTTP エラーを回避するには、ReceiveMessage リクエストの HTTP レスポンスタイムアウトが WaitTimeSeconds パラメータよりも長いことを確認してください。詳細については、「」を参照してくださいReceiveMessage。

問題のあるメッセージのキャプチャ

処理できないすべてのメッセージをキャプチャし、正確な CloudWatchメトリクスを収集するには、デッドレターキュー を設定します。

- Redrive ポリシーは、ソースキューがメッセージの処理の失敗を指定回数繰り返した後に、デッドレターキューにメッセージをリダイレクトします。
- デッドレターキューを使用するとメッセージ数が減少し、ポイズンピルメッセージ (受信されたが処理できないメッセージ) が発生する可能性が低下します。
- ポイズンピルメッセージをキューに含めると、ポイズンピルメッセージの経過時間が不正確になり、ApproximateAgeOfOldestMessage CloudWatch メトリクスが歪む可能性があります。デッドレターキューを設定すると、このメトリクスを使用する場合の誤ったアラームの回避に役立ちます。

デッドレターキュー保持の設定

スタンダードキューでは、メッセージの有効期限は、常に元のエンキューのタイムスタンプに基づきます。デッドレターキューに移動すると、エンキューのタイムスタンプは変更されません。ApproximateAgeOfOldestMessageメトリックは、メッセージが最初に送信された日ではなく、メッセージがデッドレターキューに移動した日を示します。たとえば、メッセージがデッドレターキューに移動される前に、元のキューで1日費やすと仮定します。デッドレターキューの保持期間が4日間である場合、メッセージは3日後にデッドレターキューから削除され、ApproximateAgeOfOldestMessageは3日間です。したがって、デッドレターキューの保持期間を、元のキューの保持期間よりも長く設定することがベストプラクティスです。

FIFO キューでは、メッセージがデッドレターキューに移動すると、エンキューのタイムスタンプがリセットされます。ApproximateAgeOfOldestMessage メトリクスは、メッセージがデッドレターキューに移動した日を示します。上記の同じ例では、メッセージは 4 日後にデッドレターキューから削除され、ApproximateAgeOfOldestMessage は 4 日間です。

メッセージ処理の不整合の回避

Amazon SQS は分散システムであるため、ReceiveMessage API メソッド呼び出しから正常に戻るときにAmazon SQS がメッセージを配信済みとマークしても、コンシューマーはメッセージを受信しない可能性があります。この場合、Amazon SQSはコンシューマーがメッセージを受信していない場合でも、少なくとも 1 回はメッセージを配信済みとして記録します。これらの状況ではメッセージの配信は再試行されないため、デッドレターキューの最大受信数を 1 に設定することはお勧めしません。

リクエストと応答システムの実装

リクエストと応答またはリモートプロシージャ呼び出し (RPC) システムを実行するときは、次のベストプラクティスに留意してください:

- メッセージごとに返信キューを作成しないでください。代わりに、プロデューサー ごとに起動時に返信キューを作成し、相関 ID メッセージ属性を使用して返信をリクエストにマッピングします。
- プロデューサーで返信キューを共有しないでください。共有した場合、プロデューサーは他のプロデューサー向けの応答メッセージを受信する可能性があります。

Temporary Queue Client を使用したリクエスト/応答パターンの実行の詳細については、「リクエスト–レスポンスメッセージングパターン(仮想キュー)」を参照してください。

Amazon SQSコストを削減する

次のベストプラクティスは、コストの削減や効率的なメッセージ処理に役立つだけでなく、ほぼ即時のレスポンスをもたらします。

メッセージアクションのバッチ処理

コストを削減するには、メッセージのアクションをバッチ処理します:

- メッセージを送信、受信、および削除し、1 つのアクションで複数のメッセージのメッセージ可視性タイムアウトを変更するには、Amazon SQS バッチ API アクションを使用します。
- クライアント側のバッファリングとリクエストのバッチ処理を組み合わせるには、に含まれるバッファされた非同期クライアントと AWS SDK for Java一緒にロングポーリングを使用します。

> **Note**
>
> Amazon SQS バッファリング非同期クライアントは現在 FIFOキューをサポートしていません。

適切なポーリングモードの使用

- ロングポーリングにより、Amazon SQSキューからメッセージが利用可能になるとすぐに処理することができます。

  - Amazon SQS の使用でコストを削減し、空のキューでの空の受信数 (メッセージを返さない ReceiveMessage アクションへのレスポンス) を減らすには、ロングポーリングを有効にします。詳細については、「Amazon SQS ロングポーリング」を参照してください。
  - 複数の受信により、複数のスレッドをポーリングする際の効率を高めるには、スレッド数を減らします。
  - ロングポーリングは、ほとんどの場合にショートポーリングよりも推奨されます。
- ショートポーリングは、ポーリングされたAmazon SQSキューが空の場合でも、すぐにレスポンスを返します。
  - ReceiveMessage リクエストへの即時のレスポンスが期待されるアプリケーションの要件を満たすには、ショートポーリングを使用します。
  - ショートポーリングはロングポーリングと同じように請求されます。

Amazon SQSスタンダード キューからFIFOキューへの移行

各メッセージに DelaySeconds パラメータを設定していない場合は、送信されたメッセージごとのメッセージグループ ID を指定してFIFOキューに移行できます。

詳細については、「Amazon SQS の標準キューからFIFOキューへの移行」を参照してください。

Amazon SQS FIFOキューのその他の推奨事項

次のベストプラクティスは、メッセージ重複排除 ID とメッセージグループ ID を最適に使用するのに役立ちます。詳細については、{Amazon Simple キュー Service API リファレンス}の[SendMessageとSendMessageBatchアクション]を参照してください。

トピック

- Amazon SQSメッセージ重複排除ID の使用
- Amazon SQSメッセージグループ ID の使用
- Amazon SQS受信リクエスト試行 ID の使用

Amazon SQSメッセージ重複排除ID の使用

メッセージ重複排除 ID は、送信されたメッセージの重複排除に使用されるトークンです。特定のメッセージ重複除外ID を持つメッセージが正常に送信されると、同じメッセージ重複除外ID で送信されたメッセージは正常に受け入れられますが、5分間の重複除外間には配信されません。

> **Note**
>
> Amazon SQS は、メッセージを受信して削除した後も、メッセージの重複排除 ID を追跡し続けます。

メッセージ重複排除ID の提供

プロデューサーは以下のシナリオにおいて各メッセージの重複排除ID 値を提供する必要があります:

- Amazon SQSで一意として扱う必要があるメッセージは同一メッセージ本文で送信されるメッセージ。
- Amazon SQSで一意として扱う必要がある同一の内容で異なるメッセージ属性で送信されるメッセージ。
- Amazon SQSで重複として扱う必要がある異なる内容 (たとえば、メッセージ本文に再試行回数が含まれる) で送信されるメッセージ。

シングルプロデューサー/コンシューマーシステムにおける重複排除を有効にする。

プロデューサーとコンシューマーがそれぞれ単一であり、メッセージ本文にアプリケーション固有のメッセージ ID が含まれているためメッセージが一意である場合は、次のベストプラクティスに従います。

- キューでコンテンツベースの重複排除を有効にします (メッセージ本文がそれぞれ一意)。プロデューサーはメッセージ重複排除 ID を省略できます。
- Amazon SQS FIFO キューでコンテンツベースの重複除外が有効になっていて、重複除外 ID でメッセージが送信されると、SendMessage重複除外 ID は生成されたコンテンツベースの重複除外 ID を上書きします。
- コンシューマーでは各リクエストに対する受信リクエスト試行 ID は必須ではありませんが、失敗再試行シーケンスの実行が速くなるため、これがベストプラクティスです。
- FIFOキュー内のメッセージの順序が妨げられることはないのでリクエストの送信や受信は再試行できます。

停止の復旧シナリオを考慮した設計

FIFOキューの重複排除プロセスでは、時間的制約があります。アプリケーションを設計するときは、プロデューサーとコンシューマーの両方で、クライアントのケースやネットワークが停止した場合に復旧できるようにします。

- プロデューサーは、キューの重複排除間隔を認識する必要があります。Amazon SQS の重複排除間隔は 5 分です。重複排除間隔の期限後に SendMessage リクエストを再試行すると、キューに重複メッセージが作成される場合があります。たとえば、車内からモバイルデバイスで、順番が重要なメッセージを送信するとします。確認を受信する前に車が一定時間モバイル接続を失った場合、モバイル接続が回復した後もう一度リクエストを再試行すると、重複が発生します。
- コンシューマーには、可視性タイムアウトが切れる前にメッセージを処理できなくなるリスクを最小化する可視性タイムアウトが必要です。ChangeMessageVisibility アクションをコールして、メッセージが処理されている間に可視性タイムアウトを延長できます。ただし、可視性タイムアウトの期限が切れると、別のコンシューマーがすぐにメッセージの処理を開始するため、1 つのメッセージが複数回処理されてしまいます。このシナリオを回避するには、デッドレターキューを設定します。

可視性タイムアウトの操作

最適なパフォーマンスを得るには、可視性タイムアウトを AWS SDK 読み取りタイムアウトよりも大きく設定します。これは、ショートポーリングまたはロングポーリングで使用されるReceiveMessageAPI アクションの場合に使用されます。

Amazon SQSメッセージグループ ID の使用

MessageGroupId 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タグです。同じメッセージグループに属するメッセージは、常にメッセージグループに対して厳密な順序で 1 つずつ処理されます (ただし、異なるメッセージグループに属するメッセージは、順不同に処理されます) 。

複数の順序付けされたメッセージグループのインターリーブ

単一のFIFO キューに複数の順序付けされたメッセージグループをインターリーブするには、メッセージグループ ID 値を使用します (たとえば、複数のユーザーによるセッションデータ)。このシナリオでは、複数のコンシューマーでキューを処理できますが、各ユーザーのセッションデータはFIFO方式で処理されます。

> **Note**
>
> 特定のメッセージグループ ID に属するメッセージが表示されない場合、他のコンシューマーでも同じメッセージグループ ID のメッセージを処理できません。

マルチプロデューサー-コンシューマーシステムでの重複処理の回避

順序よりもスループットやレイテンシーが重要な場合に、システム内で複数のプロデューサーおよびコンシューマーが重複メッセージを処理することを避けるには、プロデューサーがメッセージごとに一意のメッセージグループ ID を生成する必要があります。

> **Note**
>
> このシナリオでは、重複は削除されます。ただし、メッセージの順序は保証できません。複数のプロデューサーとコンシューマーを使用するシナリオでは、あるワーカーが可視性タイムアウト内にメッセージを処理せず、他のワーカーがそのメッセージを使用できるようになった場合、意図せず重複メッセージを配信するリスクは増大します。

同じメッセージグループ ID を持つメッセージの大きなバックログを回避する

FIFO キューでは、最大 20,000 通 の処理中メッセージ (コンシューマがキューから受信して、キューから削除されていないもの) が存在する可能性があります。このクォータに達した場合、Amazon SQS はエラーメッセージを返信しません。FIFOキューは、最初の 20k メッセージを検索して、使用可能なメッセージグループを判別します。つまり、1 つのメッセージグループにメッセージのバックログがある場合、バックログからメッセージを正常に消費するまで、後からキューに送信された他のメッセージグループからのメッセージを使用することはできません。

> **Note**
>
> コンシューマーがメッセージを正常に処理できないために、同じメッセージグループ ID を持つメッセージのバックログが構築される場合があります。メッセージ処理の問題は、メッセージの内容に関する問題、またはコンシューマーに関する技術的な問題が原因で発生する可能性があります。
>
> 繰り返し処理できないメッセージを移動し、同じメッセージグループ ID を持つ他のメッセージの処理のブロックを解除するには、デッドレターキューポリシーを設定することを検討してください。

仮想キューでの同じメッセージグループ ID の再使用を避ける

同じホストキューを持つ異なる仮想キューに送られた同じメッセージグループ ID を持つメッセージが互いにブロックされるのを防ぐために、仮想キューで同じメッセージグループ ID をに再使用しないようにしてください。

Amazon SQS受信リクエスト試行 ID の使用

受信リクエスト試行 ID は、重複排除に使用されるトークンです。`ReceiveMessage`を呼び出します。

SDK とAmazon SQSの間に接続性の問題が生じるような長期のネットワーク停止中は、受信リクエスト試行 ID を指定して、SDK オペレーションが失敗した場合に同じ受信リクエスト試行 ID を使用して再試行するのがベストプラクティスです。

Amazon SQS Java SDKの例

を使用して AWS SDK for Java 、Amazon Simple Queue Service (Amazon SQS) およびその他の AWS サービスとやり取りする Java アプリケーションを構築できます。SDK をインストールしてセットアップする開始方法のAWS SDK for Java 2.x デベロッパーガイドを参照してください。

キューの作成やメッセージの送信方法など、Amazon SQS キューオペレーションの基本的な操作方法例については開発ガイドの「Amazon SQS メッセージキューを使用した作業方法AWS SDK for Java 2.x」を参照してください。

このトピックの例では、サーバー側の暗号化 (SSE)、コスト配分タグ、メッセージ属性など、Amazon SQSの追加機能について説明します。

トピック

- Amazon SQS キューでのサーバー側の暗号化の使用
- Amazon SQS キューのタグの設定
- Amazon SQS キューへのメッセージ属性の送信

Amazon SQS キューでのサーバー側の暗号化の使用

を使用して AWS SDK for Java 、Amazon SQS キューにサーバー側の暗号化 (SSE) を追加できます。各キューは AWS Key Management Service （ AWS KMS) KMS キーを使用してデータ暗号化キーを生成します。この例では、Amazon SQS の AWS マネージド KMS キーを使用します。Amazon SQS SSE および KMS キーのロールの使用における詳しい情報については、「Amazon SQS での保管時の暗号化」をご参照ください。

既存のキューに SSE を追加

既存のキューのサーバーサイドの暗号化を有効にするには、`SetQueueAttributes`メソッドを使用して`KmsMasterKeyId`属性を設定します。

次のコード例では、 を Amazon SQS の AWS マネージド KMS キー AWS KMS key として設定します。 Amazon SQS また、この例では、AWS KMS key 再利用期間を 140 秒に設定します。

サンプルコードを実行する前に、 AWS 認証情報が設定されていることを確認してください。詳細については、「 AWS SDK for Java 2.x 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設定」を参照してください。

```
// Create an SqsClient for the specified Region.
SqsClient sqsClient = SqsClient.builder().region(Region.US_WEST_1).build();

// Get the URL of your queue.
String myQueueName = "my queue";
GetQueueUrlResponse getQueueUrlResponse =

 sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(myQueueName).build());
String queueUrl = getQueueUrlResponse.queueUrl();

// Create a hashmap for the attributes. Add the key alias and reuse period to the
 hashmap.
HashMap<QueueAttributeName, String> attributes = new HashMap<QueueAttributeName,
 String>();
final String kmsMasterKeyAlias = "alias/aws/sqs";  // the alias of the AWS managed KMS
 key for Amazon SQS.
attributes.put(QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias);
attributes.put(QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140");

// Create the SetQueueAttributesRequest.
SetQueueAttributesRequest set_attrs_request = SetQueueAttributesRequest.builder()
          .queueUrl(queueUrl)
          .attributes(attributes)
          .build();

sqsClient.setQueueAttributes(set_attrs_request);
```

キューのSSEの無効化

既存のキューに対してサーバーサイドの暗号化を無効にするには、SetQueueAttributes メソッドを使用して、KmsMasterKeyId属性を空の文字列に設定します。

> **Important**
>
> null は、の有効な値ではありません。KmsMasterKeyId

SSEを使用してキューを作成する

キューの作成時にSSEを有効にするには、API メソッドKmsMasterKeyIdに属性CreateQueueを追加します。

以下の例では、SSE を有効にして新しいキューを作成する方法を示します。このキューは、Amazon SQS の AWS マネージド KMS キーを使用します。また、この例では、AWS KMS key 再利用期間を 160 秒に設定します。

サンプルコードを実行する前に、 AWS 認証情報が設定されていることを確認してください。詳細については、「 AWS SDK for Java 2.x 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設定」を参照してください。

```
// Create an SqsClient for the specified Region.
SqsClient sqsClient = SqsClient.builder().region(Region.US_WEST_1).build();

// Create a hashmap for the attributes. Add the key alias and reuse period to the
 hashmap.
HashMap<QueueAttributeName, String> attributes = new HashMap<QueueAttributeName,
 String>();
final String kmsMasterKeyAlias = "alias/aws/sqs";  // the alias of the AWS managed KMS
 key for Amazon SQS.
attributes.put(QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias);
attributes.put(QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140");

// Add the attributes to the CreateQueueRequest.
CreateQueueRequest createQueueRequest =
                CreateQueueRequest.builder()
                        .queueName(queueName)
                        .attributes(attributes)
                        .build();
sqsClient.createQueue(createQueueRequest);
```

SSE属性の取得

キュー属性の取得の詳細については、Amazon Simple キューサービス API リファレンスの例をご参照ください。

特定のキューの KMS キー ID またはデータキーの再利用期間を取得する場合、GetQueueAttributes メソッドを実行して KmsMasterKeyId および KmsDataKeyReusePeriodSeconds 値を取得します。

Amazon SQS キューのタグの設定

コスト配分タグを使用して、Amazon SQSキューの整理および識別するようにします。 AWS SDK for Javaを使用してタグを設定する方法を以下の例に示します。詳細については、「Amazon SQSコスト配分タグ」を参照してください。

サンプルコードを実行する前に、 AWS 認証情報が設定されていることを確認してください。詳細については、「 AWS SDK for Java 2.x 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設定」を参照してください。

タグを一覧表にする

キューのタグを一覧表示するには、ListQueueTagsの方法を使用します。

```
// Create an SqsClient for the specified region.
SqsClient sqsClient = SqsClient.builder().region(Region.US_WEST_1).build();

// Get the queue URL.
String queueName = "MyStandardQ1";
GetQueueUrlResponse getQueueUrlResponse =

 sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
String queueUrl = getQueueUrlResponse.queueUrl();

// Create the ListQueueTagsRequest.
final ListQueueTagsRequest listQueueTagsRequest =

 ListQueueTagsRequest.builder().queueUrl(queueUrl).build();

// Retrieve the list of queue tags and print them.
final ListQueueTagsResponse listQueueTagsResponse =
                                    sqsClient.listQueueTags(listQueueTagsRequest);
System.out.println(String.format("ListQueueTags: \tTags for queue %s are %s.\n",
                queueName, listQueueTagsResponse.tags() ));
```

タグの追加または更新するには

キューのタグ値を追加または更新するには、TagQueueの方法を使用します。

```
 // Create an SqsClient for the specified Region.
```

```
SqsClient sqsClient = SqsClient.builder().region(Region.US_WEST_1).build();

// Get the queue URL.
String queueName = "MyStandardQ1";
GetQueueUrlResponse getQueueUrlResponse =

 sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
String queueUrl = getQueueUrlResponse.queueUrl();

// Build a hashmap of the tags.
final HashMap<String, String> addedTags = new HashMap<>();
        addedTags.put("Team", "Development");
        addedTags.put("Priority", "Beta");
        addedTags.put("Accounting ID", "456def");

//Create the TagQueueRequest and add them to the queue.
final TagQueueRequest tagQueueRequest = TagQueueRequest.builder()
        .queueUrl(queueUrl)
        .tags(addedTags)
        .build();
sqsClient.tagQueue(tagQueueRequest);
```

タグの削除

キューから1つ以上のタグを削除するには、UntagQueueの方法を使用します。次の例では、Accounting ID タグを 削除しています。

```

// Create the UntagQueueRequest.
final UntagQueueRequest untagQueueRequest = UntagQueueRequest.builder()
        .queueUrl(queueUrl)
        .tagKeys("Accounting ID")
        .build();

// Remove the tag from this queue.
sqsClient.untagQueue(untagQueueRequest);
```

Amazon SQS キューへのメッセージ属性の送信

メッセージ属性をメッセージに使用すると、構造化メタデータ (タイムスタンプ、地理空間データ、署名、識別子) などを指定することができます。詳細については、「Amazon SQSメッセージ属性」を参照してください。

サンプルコードを実行する前に、 AWS 認証情報が設定されていることを確認してください。詳細については、「 AWS SDK for Java 2.x 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設定」を参照してください。

属性の定義

メッセージの属性を定義するには、`MessageAttributeValue` データタイプを使用する以下のコードを追加します。詳細については、「メッセージ属性コンポーネント」および「メッセージ属性のデータ型」を参照してください。

は、メッセージ本文とメッセージ属性のチェックサム AWS SDK for Java を自動的に計算し、Amazon SQS が返すデータと比較します。詳細については、AWS SDK for Java 2.x デベロッパーガイドを参照してください。他のプログラミング言語については、「メッセージ属性のMD5メッセージダイジェストの計算」を参照してください。

String

この例では、値が`String`である、`Name`という名前の`Jane`属性を定義します。

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("Name", new MessageAttributeValue()
.withDataType("String")
.withStringValue("Jane"));
```

Number

この例では、値が`Number`である、`AccurateWeight`という名前の`230.00000000000000001`属性を定義します。

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("AccurateWeight", new MessageAttributeValue()
.withDataType("Number")
.withStringValue("230.00000000000000001"));
```

Binary

この例では、初期化されていない10バイト配列の値を持つ、Binaryという名前のByteArray属性を定義します。

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("ByteArray", new MessageAttributeValue()
.withDataType("Binary")
.withBinaryValue(ByteBuffer.wrap(new byte[10])));
```

String (custom)

この例では、値がString.EmployeeIdである、EmployeeIdという名前のカスタム属性ABC123456を定義します。

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("EmployeeId", new MessageAttributeValue()
.withDataType("String.EmployeeId")
.withStringValue("ABC123456"));
```

Number (custom)

この例では、値がNumber.AccountIdである、AccountIdという名前のカスタム属性000123456を定義します。

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("AccountId", new MessageAttributeValue()
.withDataType("Number.AccountId")
.withStringValue("000123456"));
```

> **Note**
>
> ベースデータタイプはNumberであるため、ReceiveMessageメソッドは123456を返します。

Binary (custom)

この例では、初期化されていない10バイト配列の値を持つ、Binary.JPEGという名前のカスタム属性ApplicationIconを定義します。

```
final Map<String, MessageAttributeValue> messageAttributes = new HashMap<>();
messageAttributes.put("ApplicationIcon", new MessageAttributeValue()
.withDataType("Binary.JPEG")
.withBinaryValue(ByteBuffer.wrap(new byte[10])));
```

属性を含むメッセージの送信

この例では、SendMessageRequestメッセージを送信する前に属性を追加します。

```
// Send a message with an attribute.
final SendMessageRequest sendMessageRequest = new SendMessageRequest();
sendMessageRequest.withMessageBody("This is my message text.");
sendMessageRequest.withQueueUrl(myQueueUrl);
sendMessageRequest.withMessageAttributes(messageAttributes);
sqs.sendMessage(sendMessageRequest);
```

> **Important**
>
> First-In-First-Out(FIFO)キューにメッセージを送信する場合は、メッセージグループIDを提供したsendMessage後で、 メソッドが実行されることを確認します。
> SendMessageの代わりにSendMessageBatch を使用する場合は、バッチの各メッセージに対してメッセージ属性を指定する必要があります。

Amazon SQS APIの操作

このセクションでは、Amazon SQS エンドポイントの構築、GET メソッドおよび POST メソッドを使用したクエリ API リクエストの作成、およびバッチ API アクションの使用に関する情報を提供します。パラメータ、エラー、例、およびデータ型を含む Amazon SQS アクションの詳細については、「Amazon Simple Queue Service API リファレンス」を参照してください。

さまざまなプログラミング言語を使用して Amazon SQS にアクセスするには、以下の自動機能が含まれた AWS SDK を使用することもできます。

- サービスリクエストに暗号署名する
- リクエストを再試行する
- エラーレスポンスの処理をする

コマンドラインツール情報については、「AWS CLI コマンドリファレンス」の「Amazon SQS セクション」、および「AWS Tools for PowerShell Cmdlet リファレンス」を参照してください。

AWS JSON プロトコルを使用した Amazon SQS APIs

Amazon SQS は、指定された SDK バージョン のすべての Amazon SQS APIs のトランスポートメカニズムとして AWS JSON プロトコルを使用します。 AWS JSON プロトコルは、 AWS クエリプロトコルと比較して、より高いスループット、低レイテンシー、および高速な application-to-application 通信を提供します。 AWS JSON プロトコルは、リクエストとレスポンスのシリアル化/逆シリアル化をより効率的にします。 AWSそれでも SQS API で AWS クエリプロトコルを使用する場合は、Amazon SQS クエリプロトコルをサポートする AWS SDK バージョンAmazon SQS API で使用される AWS JSON プロトコルではどの言語がサポートされていますか。については、「」を参照してください。 APIs Amazon SQS AWS

Amazon SQS は、 AWS JSON プロトコルを使用して AWS SDK クライアント (Java、Python、Golang など JavaScript) と Amazon SQS サーバー間の通信を行います。Amazon SQS API オペレーションの HTTP リクエストは、JSON 形式の入力を受け付けます。Amazon SQS オペレーションが実行され、実行レスポンスが JSON 形式で SDK クライアントに送り返されます。AWS クエリと比較して、 AWS JSON はクライアントとサーバー間でデータを転送するために、より簡単、迅速、効率的になります。

- AWS JSON プロトコルは、Amazon SQS クライアントとサーバーの間の仲介役として機能します。

- サーバーは Amazon SQS オペレーションが作成されるプログラミング言語を理解していませんが、 AWS JSON プロトコルを理解しています。
- AWS JSON プロトコルは、Amazon SQS クライアントとサーバー間のシリアル化 (オブジェクトを JSON 形式に変換) とシリアル化解除 (JSON 形式をオブジェクトに変換) を使用します。

Amazon SQS を使用した AWS JSON プロトコルの詳細については、「」を参照してくださいAmazon SQS AWS JSON プロトコルFAQs。 Amazon SQS

AWS JSON プロトコルは、指定された AWS SDK バージョン で使用できます。さまざまな言語バージョンの SDK バージョンとリリース日を確認するには、「AWS SDK およびツールリファレンスガイド」の「AWS SDK とツールのバージョンサポートマトリックス」を参照してください。

トピック

- Amazon SQS で AWS JSON プロトコルを使用してクエリ API リクエストを実行する Amazon SQS
- Amazon SQS でのクエリプロトコルを使用した AWS クエリ API リクエストの実行 Amazon SQS
- Amazon SQS のリクエストの認証
- Amazon SQSのバッチアクション

Amazon SQS で AWS JSON プロトコルを使用してクエリ API リクエストを実行する Amazon SQS

このセクションでは、Amazon SQS エンドポイントを構築する方法、POST リクエストを行う方法、およびレスポンスを解釈する方法について説明します。

> **Note**
>
> AWS JSON プロトコルは、ほとんどの言語バリアントでサポートされています。サポートされている言語のリストについては、「Amazon SQS API で使用される AWS JSON プロトコルではどの言語がサポートされていますか。」を参照してください。

トピック

- エンドポイントの構築
- POSTリクエストの作成

- Amazon SQS JSON API レスポンスの解釈
- Amazon SQS AWS JSON プロトコルFAQs

エンドポイントの構築

Amazon SQS キューを使用するには、エンドポイントを構築する必要があります。Amazon SQS エンドポイントの詳細については、「Amazon Web Services 全般のリファレンス」の以下のページを参照してください。

- リージョンエンドポイント
- Amazon Simple Queue Serviceエンドポイントとクォータ

Amazon SQSエンドポイントはそれぞれ独立しています。例えば、2 つのキューに という名前が付けられMyQueueていて、一方に エンドポイントsqs.us-east-2.amazonaws.comがあり、もう一方のキューに エンドポイントがある場合sqs.eu-west-2.amazonaws.com、2 つのキューは相互にデータを共有しません。

キュー作成のクエストを行うエンドポイントの例を次に示します。

```
POST / HTTP/1.1
Host: sqs.us-west-2.amazonaws.com
X-Amz-Target: AmazonSQS.CreateQueue
X-Amz-Date: <Date>
Content-Type: application/x-amz-json-1.0
Authorization: <AuthParams>
Content-Length: <PayloadSizeBytes>
Connection: Keep-Alive
{
    "QueueName":"MyQueue",
    "Attributes": {
        "VisibilityTimeout": "40"
    },
    "tags": {
        "QueueType": "Production"
    }
}
```

> **Note**
>
> キュー名とキュー URL では、大文字と小文字が区別されます。
> *AUTHPARAMS*の構造はAPIリクエストの署名によって異なります。詳細については、Amazon Web Services 全般のリファレンスのAWS 「 API リクエストの署名」を参照してください。

POSTリクエストの作成

Amazon SQS の POST リクエストは、クエリパラメータを HTTP リクエストボディの形で送信します。

以下は、`X-Amz-Target` が `AmazonSQS.<operationName>` に設定された HTTP ヘッダーと、`Content-Type` が `application/x-amz-json-1.0` に設定された HTTP ヘッダーの例です。

```
POST / HTTP/1.1
Host: sqs.<region>.<domain>
X-Amz-Target: AmazonSQS.SendMessage
X-Amz-Date: <Date>
Content-Type: application/x-amz-json-1.0
Authorization: <AuthParams>
Content-Length: <PayloadSizeBytes>
Connection: Keep-Alive
{
    "QueueUrl": "https://sqs.<region>.<domain>/<awsAccountId>/<queueName>/",
    "MessageBody": "This is a test message",
}
```

この HTTP POST リクエストは、Amazon SQS キューにメッセージを送信します。

> **Note**
>
> HTTP ヘッダー `X-Amz-Target` と `Content-Type` は両方とも必須です。
> HTTPクライアントは、クライアントのHTTP バージョンによっては、他の項目を HTTPリクエストに追加する可能性があります。

Amazon SQS JSON API レスポンスの解釈

アクションリクエストに応答して、Amazon SQS はリクエスト結果を含む JSON データ構造を返します。詳細については、「Amazon Simple Queue Service API リファレンス」および「Amazon SQS AWS JSON プロトコルFAQs」の個々のアクションを参照してください。

トピック

正常な JSON レスポンス構造

リクエストが成功した場合、主なレスポンス要素は x-amzn-RequestId です。これには、リクエストの汎用一意識別子 (UUID) と、その他の付加されたレスポンスフィールドが含まれます。例えば、CreateQueue レスポンスには QueueUrl フィールドが含まれています。そのフィールドには、作成されたキューの URL が含まれています。

```
HTTP/1.1 200 OK
x-amzn-RequestId: <requestId>
Content-Length: <PayloadSizeBytes>
Date: <Date>
Content-Type: application/x-amz-json-1.0
{
    "QueueUrl":"https://sqs.us-east-1.amazonaws.com/111122223333/MyQueue"
}
```

JSON エラーレスポンス構造

リクエストが正常に行われなかった場合、Amazon SQS は HTTP ヘッダーとボディを含む主なレスポンスを返信します。

HTTP ヘッダーで、x-amzn-RequestId にはリクエストの UUID が含まれています。x-amzn-query-error には、エラーのタイプと、エラーがプロデューサーエラーかコンシューマーエラーかという 2 つの情報が含まれています。

レスポンスボディで、"__type" はその他のエラーの詳細を示し、Message は読みやすい形式でエラー状況を示します。

以下は、JSON 形式のエラーレスポンスの例です。

```
HTTP/1.1 400 Bad Request
x-amzn-RequestId: 66916324-67ca-54bb-a410-3f567a7a0571
x-amzn-query-error: AWS.SimpleQueueService.NonExistentQueue;Sender
Content-Length: <PayloadSizeBytes>
Date: <Date>
Content-Type: application/x-amz-json-1.0
{
    "__type": "com.amazonaws.sqs#QueueDoesNotExist",
    "message": "The specified queue does not exist."
}
```

Amazon SQS AWS JSON プロトコルFAQs

Amazon SQS での AWS JSON プロトコルの使用に関するよくある質問。 Amazon SQS

AWS JSON プロトコルとは何ですか。また、既存の Amazon SQS API リクエストとレスポンスとどのように異なりますか？

JSON は、異種システム間の通信で最も広く使用され、受け入れられている配線方法の 1 つです。Amazon SQS は JSON をメディアとして使用して、 AWS SDK クライアント (Java、Python、Golang など JavaScript) と Amazon SQS サーバー間の通信を行います。Amazon SQS API オペレーションの HTTP リクエストは、JSON 形式の入力を受け付けます。Amazon SQS オペレーションが実行され、実行レスポンスが JSON 形式で SDK クライアントに共有されます。AWS クエリと比較して、JSON はクライアントとサーバー間でより効率的にデータ転送できます。

- Amazon SQS AWS JSON プロトコルは、Amazon SQS クライアントとサーバーの間の仲介役として機能します。
- サーバーは Amazon SQS オペレーションが作成されるプログラミング言語を理解していませんが、 AWS JSON プロトコルを理解しています。
- Amazon SQS AWS JSON プロトコルは、Amazon SQS クライアントとサーバー間のシリアル化 (オブジェクトを JSON 形式に変換) と逆シリアル化 (JSON 形式をオブジェクトに変換) を使用します。

Amazon SQS の AWS JSON プロトコルの使用を開始するにはどうすればよいですか? Amazon SQS

Amazon SQS のメッセージングを高速化するために最新の AWS SDK バージョンの使用を開始するには、 AWS SDK を指定されたバージョンまたはそれ以降のバージョンにアップグレードします。 Amazon SQS SDK クライアントの詳細については、以下の表の「ガイド」列を参照してください。

以下は、Amazon SQS API で使用する AWS JSON プロトコルの言語バリアント間の SDK バージョンのリストです。 Amazon SQS APIs

[言語]	SDK クライアントリポジトリ	必要な SDK クライアントバージョン	ガイド
C++	aws/aws-sdk-cpp	1.11.98	AWS SDK for C++
Golang 1.x	aws/aws-sdk-go	v1.47.7	AWS SDK for Go
Golang 2.x	aws/aws-sdk-go-v2	v1.28.0	AWS SDK for Go V2
Java 1.x	aws/aws-sdk-java	1.12.585	AWS SDK for Java
Java 2.x	aws/aws-sdk-java-v2	2.21.19	AWS SDK for Java
JavaScript v2.x	aws/aws-sdk-js	v2.1492.0	JavaScript 上の AWS
JavaScript v3.x	aws/aws-sdk-js-v3	v3.447.0	JavaScript 上の AWS
.NET	aws/aws-sdk-net	3.7.681.0	AWS SDK for .NET
PHP	aws/aws-sdk-php	3.285.2	AWS SDK for PHP
Python-boto3	boto/boto3	1.28.82	AWS SDK for Python (Boto3)

[言語]	SDK クライアントリポジトリ	必要な SDK クライアントバージョン	ガイド
Python-botocore	boto/botocore	1.31.82	AWS SDK for Python (Boto3)
awscli	AWS CLI	1.29.82	AWSコマンドラインインターフェイス
Ruby	aws/aws-sdk-ruby	1.67.0	AWS SDK for Ruby

Amazon SQS ワークロードで JSON プロトコルを有効にすることにはどのようなリスクがありますか。

AWS SDK のカスタム実装、またはカスタムクライアントと AWS SDK の組み合わせを使用して、AWS クエリベースの (XML ベースの) レスポンスを生成する Amazon SQS とやり取りする場合、AWS JSON プロトコルと互換性がない可能性があります。問題が発生した場合は、 AWS サポートにお問い合わせください。

すでに最新の AWS SDK バージョンを使用しているが、オープンソースソリューションが JSON をサポートしていない場合はどうなりますか？

SDK のバージョンを、使用中のバージョンより前のバージョンに変更する必要があります。詳細については[Amazon SQS の AWS JSON プロトコルの使用を開始するにはどうすればよいですか? Amazon SQS、「」を参照してください。「」に記載されている AWS SDK バージョン](Amazon SQS の AWS JSON プロトコルの使用を開始するにはどうすればよいですか? Amazon SQS)[Amazon SQS の AWS JSON プロトコルの使用を開始するにはどうすればよいですか? Amazon SQS](Amazon SQS の AWS JSON プロトコルの使用を開始するにはどうすればよいですか? Amazon SQS)は、Amazon SQS API の JSON ワイヤプロトコルを使用します。 APIs AWS SDK を以前のバージョンに変更すると、Amazon SQS APIs は AWS クエリを使用します。

Amazon SQS API で使用される AWS JSON プロトコルではどの言語がサポートされていますか。

Amazon SQS は、 AWS SDKsされているすべての言語バリアント (GA) をサポートしています。現在、Kotlin、Rust、Swift はサポートしていません。他の言語バリアントについて詳しくは、「[AWS での構築ツール」を参照してください。](AWS での構築ツール)

Amazon SQS API で使用される AWS JSON プロトコルではどのリージョンがサポートされていますか。

Amazon SQS は、Amazon SQS が利用可能なすべてのAWS リージョンで AWS JSON プロトコルをサポートしています。 Amazon SQS

AWS JSON プロトコルを使用して Amazon SQS の指定された AWS SDK バージョンにアップグレードすると、どのようなレイテンシーの改善が期待できますか? Amazon SQS

AWS JSON プロトコルは、 AWS クエリプロトコルと比較して、リクエストとレスポンスのシリアル化と逆シリアル化がより効率的です。5 KB のメッセージペイロード AWS のパフォーマンステストに基づいて、Amazon SQS の JSON プロトコルは、 end-to-end メッセージ処理のレイテンシーを最大 23% 削減し、アプリケーションクライアント側の CPU とメモリの使用量を削減します。

AWS クエリプロトコルは廃止されますか ?

AWS クエリプロトコルは引き続きサポートされます。 AWS SDK バージョンが、「Amazon SQS の JSON プロトコルの開始方法」に記載されているバージョン以外の以前のバージョンに設定されている限り、 AWS クエリプロトコルを引き続き使用できます。 AWS Amazon SQS

AWS JSON プロトコルの詳細情報はどこで入手できますか。

JSON プロトコルの詳細については、Smithy ドキュメントの「AWS JSON 1.0 プロトコル」を参照してください。 AWS JSON プロトコルを使用する Amazon SQS API リクエストの詳細については、「Amazon SQS で AWS JSON プロトコルを使用してクエリ API リクエストを実行する Amazon SQS」を参照してください。

Amazon SQS でのクエリプロトコルを使用した AWS クエリ API リクエストの実行 Amazon SQS

このセクションでは、Amazon SQS エンドポイントを構築し、GET および POST リクエストを行い、レスポンスを解釈する方法について説明します。

トピック

- エンドポイントの構築
- GETリクエストの作成
- POSTリクエストの作成

- Amazon SQS XML API レスポンスの解釈

エンドポイントの構築

Amazon SQSキューを使用するには、エンドポイントを構築する必要があります。Amazon SQS エンドポイントの詳細については、「Amazon Web Services 全般のリファレンス」の以下のページを参照してください。

- リージョンエンドポイント
- Amazon Simple Queue Serviceエンドポイントとクォータ

Amazon SQSエンドポイントはそれぞれ独立しています。例えば、2 つのキューに という名前が付けられMyQueue、一方にエンドポイント sqs.us-east-2.amazonaws.com があり、もう一方にエンドポイント がある場合sqs.eu-west-2.amazonaws.com、2 つのキューは互いにデータを共有しません。

キューを作成するリクエストを行うエンドポイントの例を次に示します。

```
https://sqs.eu-west-2.amazonaws.com/
?Action=CreateQueue
&DefaultVisibilityTimeout=40
&QueueName=MyQueue
&Version=2012-11-05
&AUTHPARAMS
```

> **Note**
>
> キュー名とキュー URL では、大文字と小文字が区別されます。
> *AUTHPARAMS*の構造はAPIリクエストの署名によって異なります。詳細については、Amazon Web Services 全般のリファレンスのAWS 「 API リクエストの署名」を参照してください。

GETリクエストの作成

Amazon SQS GET リクエストは、以下で構成される URL として構造化されます。

- エンドポイント-リクエストが作用するリソース (キュー名と URL)。たとえば: `https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue`

- アクション-エンドポイントに実行したい アクション。疑問符 (?) によってエンドポイントとアクションが区切られます。たとえば?Action=SendMessage&MessageBody=Your%20Message%20Textなど:
- Parameters – 任意のリクエストパラメータ。各パラメータは、アンパサンド (&) で区切られます。例: &Version=2012-11-05&*AUTHPARAMS*

以下は、Amazon SQS キューにメッセージを送信する GET リクエストの例です。

```
https://sqs.us-east-2.amazonaws.com/123456789012/MyQueue
?Action=SendMessage&MessageBody=Your%20message%20text
&Version=2012-11-05
&AUTHPARAMS
```

> **Note**
>
> キュー名とキュー URL では、大文字と小文字が区別されます。
> GET リクエストは URLsであるため、すべてのパラメータ値を URL エンコードする必要があります。URLにはスペースを使用できないため、各スペースは「%20」としてURLエンコードされます。この例の残りの部分は、読みやすくするために URL エンコードされていません。

POSTリクエストの作成

Amazon SQS の POST リクエストは、クエリパラメータを HTTP リクエストボディの形で送信します。

以下は、 が Content-Typeに設定されている HTTP ヘッダーの例ですapplication/x-www-form-urlencoded。

```
POST /123456789012/MyQueue HTTP/1.1
Host: sqs.us-east-2.amazonaws.com
Content-Type: application/x-www-form-urlencoded
```

ヘッダーの後に、Amazon SQS キューにメッセージを送信する form-urlencoded GET リクエストが続きます。 Amazon SQS 各パラメータは、アンパサンド (&)で区切られています。

```
Action=SendMessage
```

```
&MessageBody=Your+Message+Text
&Expires=2020-10-15T12%3A00%3A00Z
&Version=2012-11-05
&AUTHPARAMS
```

> **Note**
>
> Content-TypeHTTPヘッダーのみが必須です。*AUTHPARAMS* は GET リクエストの場合と同じです。
> HTTPクライアントは、クライアントのHTTP バージョンによっては、他の項目を HTTPリクエストに追加する可能性があります。

Amazon SQS XML API レスポンスの解釈

アクションリクエストに応答して、Amazon SQSはリクエスト結果を含むXMLデータ構造を返信します。詳細については、「Amazon Simple Notification Service APIリファレンス」の「キューアクション」を参照してください。

トピック

- 成功した XML レスポンス構造
- XML エラーレスポンス構造

成功した XML レスポンス構造

リクエストが成功した場合、メインレスポンス要素は アクションにちなんで名前が付けられ、Response が追加されます (例: *ActionName*Response) 。

この要素には、以下の子要素が含まれています。

- ***ActionName*Result**-アクション固有の要素が含まれています。たとえば、CreateQueueResult 要素には QueueUrl 要素が含まれています。つまり、作成されたキューのURLが含まれています。
- **ResponseMetadata** – リクエストのユニバーサル一意識別子 (UUID) RequestIdを含む が含まれます。

以下に、XML形式の正常なレスポンスの例を示します。

```
<CreateQueueResponse
   xmlns=https://sqs.us-east-2.amazonaws.com/doc/2012-11-05/
   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
   xsi:type=CreateQueueResponse>
   <CreateQueueResult>
      <QueueUrl>https://sqs.us-east-2.amazonaws.com/770098461991/queue2</QueueUrl>
   </CreateQueueResult>
   <ResponseMetadata>
      <RequestId>cb919c0a-9bce-4afe-9b48-9bdf2412bb67</RequestId>
   </ResponseMetadata>
</CreateQueueResponse>
```

XML エラーレスポンス構造

リクエストが正常に行われなかった場合、 Amazon SQSは常に主な応答要素を返信しますErrorResponse。この要素には Error 要素と RequestId 要素が含まれます。

Error 要素には、以下の子要素が含まれています。

- **Type**-エラーがプロデューサーかコンシューマーのどちらかを特定します。
- **Code**-エラーのタイプを特定します。
- **Message**-読み取り可能な条件でエラー状況を特定します。
- **Detail**-(オプション) エラーに関する追加の詳細を特定します。

RequestId要素には、リクエストのUUIDが含まれています。

以下に、XML形式のエラーレスポンスの例を示します。

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

Amazon SQS のリクエストの認証

署名認証とは、リクエストの送信者を特定し、検証するプロセスです。認証の最初の段階で、 はプロデューサーの ID と、プロデューサーが を使用するように登録 AWSされているかどうかを確認します (詳細については、 AWS 「」を参照してくださいステップ 1: AWS アカウント と IAM ユーザーを作成する) 。次に、以下の手順 AWS に従います。

1. プロデューサー (送信者) が必要な証明書を取得します。
2. プロデューサーがリクエストと認証情報をコンシューマー (受信者) に送信します。
3. コンシューマーは証明書を使用して、プロデューサーが本当にリクエストを送信したか検証します。
4. 次のいずれかの結果になります。
   - 認証が成功すると、コンシューマーはリクエストを処理します。
   - 認証に失敗すると、コンシューマーによってリクエストが却下され、エラーが返されます。

トピック

- HMAC-SHA による基本的な認証のプロセス
- パート 1: ユーザーからのリクエスト
- パート 2: からのレスポンス AWS

HMAC-SHA による基本的な認証のプロセス

クエリAPIを使用してAmazon SQSにアクセスする場合は、リクエストの認証のために、以下の項目を準備する必要があります。

- を識別するAWS アクセスキー ID。シークレットアクセスキーの検索に AWS アカウント AWS を使用します。
- HMAC-SHAリクエスト署名、シークレットアクセスキーを使用して計算されます(共有シークレットはお客様と AWSだけが知っています。詳細については、RFC2104 を参照してください)。AWS SDK によって署名プロセスが処理されますが、HTTP または HTTPS 経由でクエリリクエストを送信した場合、各クエリリクエストに署名を含める必要があります。

  1. 署名バージョン4の署名キーを取得します。詳細については、「Java を使用して署名キーを取得」を参照してください。

> **Note**
>
> Amazon SQSは署名バージョン4をサポートしています。署名バージョン4では、SHA256により以前のバージョンよりもセキュリティとパフォーマンスが向上しています。Amazon SQSを使用する新規のアプリケーションを作成する場合、署名バージョン4を使用します。

2. リクエスト署名をBase64 エンコードします。以下のサンプルJavaコードによってこの処理が行われます。

```
package amazon.webservices.common;

// Define common routines for encoding data in AWS requests.
public class Encoding {

    /* Perform base64 encoding of input bytes.
     * rawData is the array of bytes to be encoded.
     * return is the base64-encoded string representation of rawData.
     */
    public static String EncodeBase64(byte[] rawData) {
        return Base64.encodeBytes(rawData);
    }
}
```

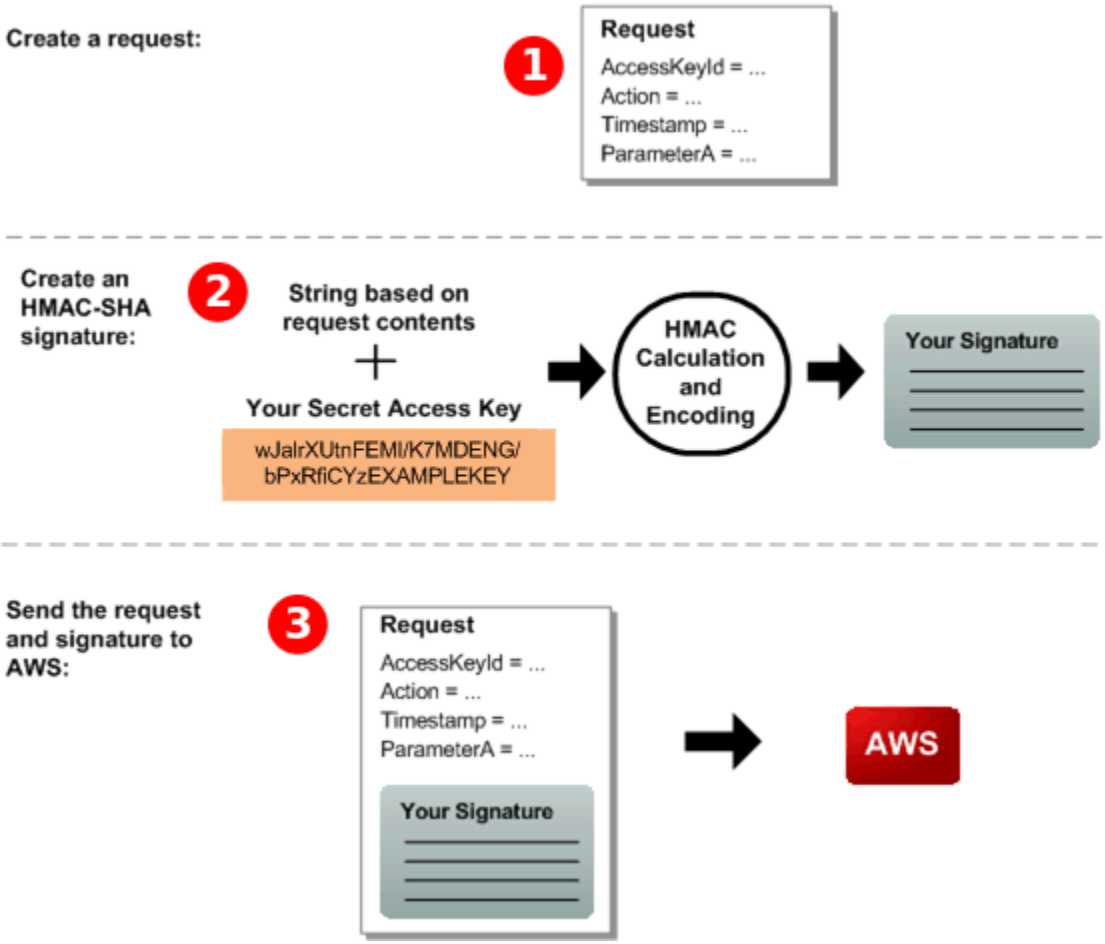
- リクエストのタイムスタンプ (または有効期限)。リクエストで使用するタイムスタンプは、dateTime オブジェクトでなければなりません。時、分、秒を含む詳細な日時が必要です。たとえば、2007-01-31T23:59:59Z などです。必須ではありませんが、協定世界時 (グリニッジ標準時)を使用してオブジェクトを提供することが推奨されています。

> **Note**
>
> あなたのサーバー時刻が正しく設定されていることを確認してください。(有効期限ではなく) タイムスタンプを指定すると、リクエストは指定された時刻から 15 AWS 分後に自動的に期限切れになります (サーバー上の AWS 現在の時刻より 15 分以上前のタイムスタンプを持つリクエストは処理されません) 。
> .NET を使用する場合、過度に厳密なタイムスタンプ情報を送信しないようご注意ください (タイムスタンプの処理プロセスの違いにより不具合が発生する可能性があるためで

す)。この場合、精度が 1ミリ秒以内のdateTimeオブジェクトを手動で作成してください。

パート 1: ユーザーからのリクエスト

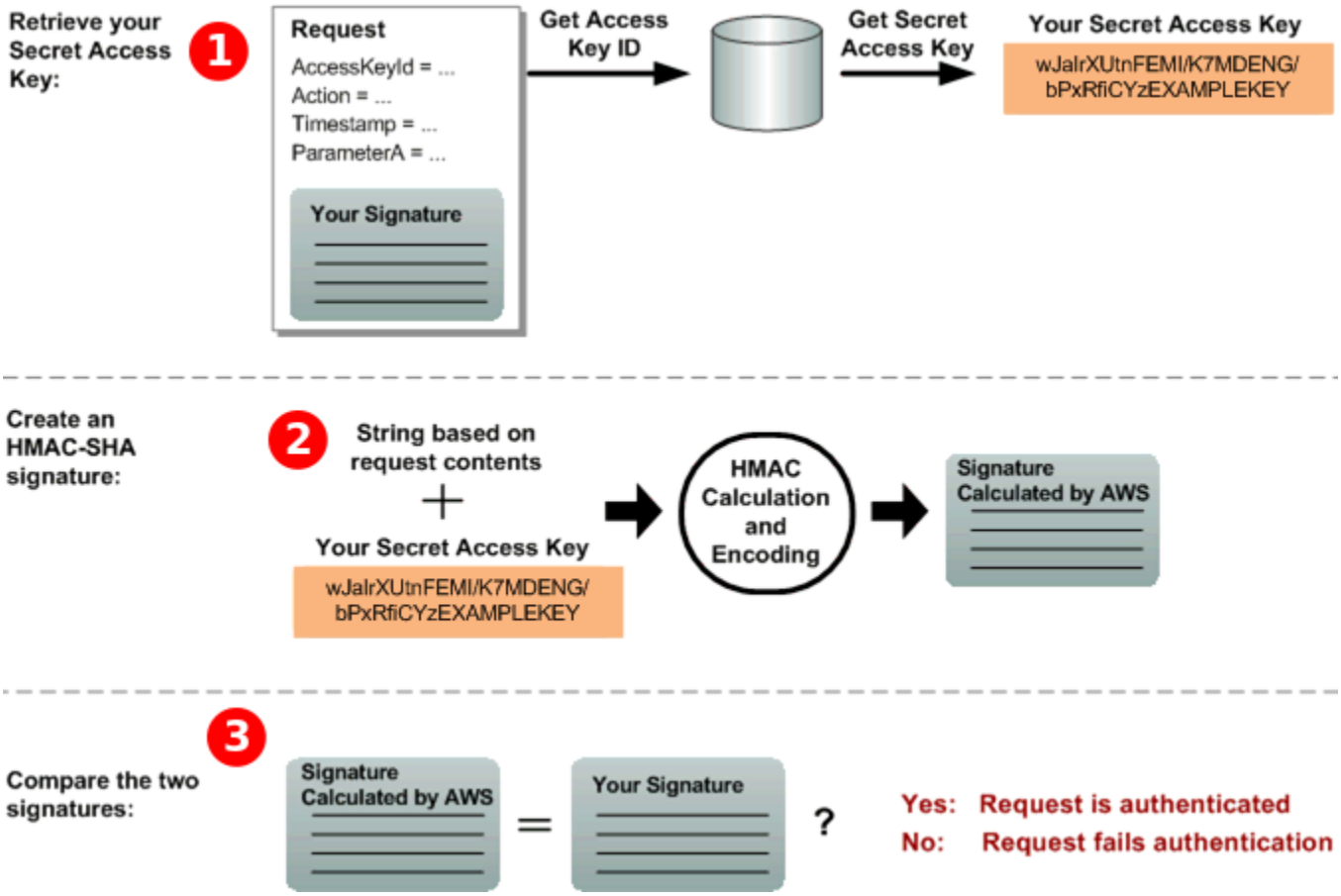
以下は、HMAC-SHA AWS リクエスト署名を使用してリクエストを認証するために従う必要があるプロセスです。



1. へのリクエストを作成します AWS。
2. シークレットアクセスキーを使用して、キー付きハッシュメッセージ認証コード (HMAC-SHA) 署名を計算します。
3. リクエストに署名とアクセスキー ID を含めてから、リクエストを に送信します AWS。

パート 2: からのレスポンス AWS

AWS は応答として次のプロセスを開始します。



1. AWS はアクセスキー ID を使用してシークレットアクセスキーを検索します。
2. AWS は、リクエストで送信した署名の計算に使用したのと同じアルゴリズムを使用して、リクエストデータとシークレットアクセスキーから署名を生成します。
3. 次のいずれかの結果になります。
   - が AWS 生成する署名がリクエストで送信した署名と一致する場合、 はリクエストが本物である AWS と見なします。
   - 比較が失敗すると、リクエストは破棄され、エラー AWS が返されます。

Amazon SQSのバッチアクション

コストを削減したり、1 回のアクションで最大 10 通のメッセージを操作したりするには、次のアクションを使用します。

- SendMessageBatch
- DeleteMessageBatch
- ChangeMessageVisibilityBatch

クエリ API または Amazon SQS バッチアクションをサポートする AWS SDK を使用して、バッチ機能を利用できます。 Amazon SQS

> **Note**
>
> 1 回のSendMessageBatch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 KiB) を超えることはできません。
> SendMessageBatch、DeleteMessageBatch、または ChangeMessageVisibilityBatch のアクセス許可を明示的に設定することはできません。SendMessage、DeleteMessage、または ChangeMessageVisibility のアクセス許可を設定すると、アクションの対応するバッチバージョンのアクセス許可も設定されます。
> Amazon SQSコンソールは、バッチアクションをサポートしていません。

トピック

- Amazon SQS によるクライアント側のバッファリングとリクエストのバッチ処理の有効化
- Amazon SQS での水平スケーリングとアクションバッチ処理を使用したスループットの向上

Amazon SQS によるクライアント側のバッファリングとリクエストのバッチ処理の有効化

AWS SDK for Javaを含むAmazonSQSBufferedAsyncClientAmazon SQSにアクセスするもの。このクライアントでは、クライアント側でのバッファリングを使用してリクエストを簡単に行うことができます。この場合、最初にクライアントからの呼び出しをバッファして、Amazon SQSへのバッチリクエストとして送信されます。

クライアント側のバッファリングは最大10個のリクエストをバッファリングし、バッチリクエストとして送信できるため、Amazon SQSの利用コストを削減して、送信リクエストの数を減らすことができます。バッファリングは同期および非同期コールの両方をAmazonSQSBufferedAsyncClient バッファします。バッチ処理されたリクエストとロングポーリングのサポートによって、スループットを向上させることもできます。詳細については、

「Amazon SQS での水平スケーリングとアクションバッチ処理を使用したスループットの向上」を参照してください。

AmazonSQSBufferedAsyncClientはAmazonSQSAsyncClientと同じインターフェイスを実行するため、AmazonSQSAsyncClient から AmazonSQSBufferedAsyncClient に移行するには通常既存のコードを最小限変更するだけです。

> **Note**
>
> Amazon SQSバッファリング非同期クライアントは現在 FIFOキューをサポートしていません。

トピック

- AmazonSQSBufferedAsync クライアントの使用
- AmazonSQSBufferedAsync クライアントの設定

AmazonSQSBufferedAsync クライアントの使用

開始する前に、「Amazon SQSのセットアップ」のステップを完了します。

> **Important**
>
> は現在 AWS SDK for Java 2.x 、 と互換性がありませんAmazonSQSBufferedAsyncClient。

たとえば、AmazonSQSAsyncClient をベースに新しい AmazonSQSBufferedAsyncClient を作成できます。

```
// Create the basic Amazon SQS async client
final AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient();

// Create the buffered client
final AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync);
```

新規作成したらAmazonSQSBufferedAsyncClient、これを使用してAmazon SQSに複数のリクエストを送信できます (AmazonSQSAsyncClient と同様です)。次に例を示します。

```
final CreateQueueRequest createRequest = new
 CreateQueueRequest().withQueueName("MyQueue");

final CreateQueueResult res = bufferedSqs.createQueue(createRequest);

final SendMessageRequest request = new SendMessageRequest();
final String body = "Your message text" + System.currentTimeMillis();
request.setMessageBody( body );
request.setQueueUrl(res.getQueueUrl());

final Future<SendMessageResult> sendResult = bufferedSqs.sendMessageAsync(request);

final ReceiveMessageRequest receiveRq = new ReceiveMessageRequest()
    .withMaxNumberOfMessages(1)
    .withQueueUrl(queueUrl);
final ReceiveMessageResult rx = bufferedSqs.receiveMessage(receiveRq);
```

AmazonSQSBufferedAsync クライアントの設定

AmazonSQSBufferedAsyncClient は、ほとんどのユースケースに合うように事前に設定されています。たとえば、AmazonSQSBufferedAsyncClient をさらに設定できます。'

1. 必要な設定パラメータを使用して、QueueBufferConfig クラスのインスタンスを作成します。
2. AmazonSQSBufferedAsyncClient コンストラクタにインスタンスを指定します。

```
// Create the basic Amazon SQS async client
final AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient();

final QueueBufferConfig config = new QueueBufferConfig()
    .withMaxInflightReceiveBatches(5)
    .withMaxDoneReceiveBatches(15);

// Create the buffered client
final AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync, config);
```

QueueBufferConfig 設定パラメータ

パラメータ	デフォルト値	説明
longPoll	true	longPoll が true に設定されている場合、AmazonSQSBufferedAsyncClient はメッセージの処理時にロングポーリングを使用しようとします。
longPollWaitTimeoutSeconds	20 s	空の受信結果を返すまでに、キュー内へのメッセージの出現をサーバーが待機するのを ReceiveMessage 呼び出しがブロックする最大秒数。 **Note** ロングポーリングが無効になっている場合、この設定に効果はありません。
maxBatchOpenMs	200ms	送信呼び出しが、同じタイプのメッセージをバッチ処理する他の呼び出しを待機する最大ミリ秒。 設定を大きくすればするほど、同じ量の処理を実行するのに必要なバッチが少なくなります (ただし、バッチ内の最初の呼び出しは待機時間が長くなります)。

パラメータ	デフォルト値	説明
		このパラメータを 0 に設定した場合、送信されたリクエストは他のリクエストを待機しないため、バッチ処理が事実上無効になります。
maxBatchSize	バッチあたり 10 個のリクエスト	1 つのバッチリクエストでまとめてバッチ処理されるメッセージの最大数。設定を大きくするほど、全体数が同じリクエストの処理に要するバッチ数が減ります。 **Note** バッチあたり 10 個のリクエストはAmazon SQSの最大許容値です。
maxBatchSizeBytes	256 KiB	クライアントがAmazon SQSに送信しようとするメッセージバッチの最大サイズ、バイト単位。 **Note** 256 KiB は Amazon SQS の最大許容値です。

パラメータ	デフォルト値	説明
maxDoneReceiveBatches	10 個のバッチ	AmazonSQSBufferedAsyncClient がプリフェッチし、クライアント側に保存する受信バッチの最大数。 設定を大きくすればするほど、Amazon SQSを呼び出さなくても多くの受信リクエストを満たすことができます (ただし、プリフェッチされるメッセージが多くなるほど、バッファにとどまる時間が長くなるため、それ自体の可視性タイムアウトが発生する可能性があります)。 **Note** 0 は、すべてのメッセージのプリフェッチが無効になっており、メッセージがオンデマンドでのみ消費されることを示します。

パラメータ	デフォルト値	説明
maxInflightOutboundBatches	5 個のバッチ	同時に処理できるアクティブな送信バッチの最大数。 設定を大きくすればするほど、送信バッチの送信速度が速くなり (CPU や帯域幅などの他のクォータの影響を受けます)、AmazonSQSBufferedAsyncClient により消費されるスレッドが増えます。
maxInflightReceiveBatches	10 個のバッチ	同時に処理できるアクティブな受信バッチの最大数。 設定を大きくすればするほど、受信するメッセージが増え (CPU や帯域幅などの他のクォータの影響を受けます)、AmazonSQSBufferedAsyncClient により消費されるスレッドが増えます。 **Note** 0 は、すべてのメッセージのプリフェッチが無効になっており、メッセージがオンデマンドでのみ消費されることを示します。

パラメータ	デフォルト値	説明
visibilityTimeoutSeconds	-1	このパラメータが 0 以外の正の値に設定されている場合、ここで設定した可視性タイムアウトにより、メッセージの処理元のキューで設定された可視性タイムアウトが上書きされます。 **Note** -1 は、キューのデフォルト設定が選択されていることを示します。 可視性タイムアウトを 0 に設定することはできません。

Amazon SQS での水平スケーリングとアクションバッチ処理を使用したスループットの向上

Amazon SQS キューにより、非常に高いスループットを実現できます。スループットクォータの詳細については、「Amazon SQS メッセージクォータ」を参照してください。

高いスループットを達成するには、メッセージのプロデューサーとコンシューマーを水平にスケーリングする必要があります (プロデューサーとコンシューマーを追加します)。

トピック

- 水平スケーリング
- アクションバッチ処理
- 1 回のオペレーションおよびバッチリクエストでの Java の使用例

水平スケーリング

Amazon SQSは HTTP リクエストレスポンスプロトコルを通じてアクセスするため、リクエストのレイテンシー (リクエストの開始からレスポンスの受信までの時間) により1回の接続を使用して1つのスレッドを処理した場合のスループットは制限されます。たとえば、Amazon EC2ベースのクライアントから同じリージョンにあるAmazon SQSへのレイテンシーが平均 20 ミリ秒の場合、1回の接続で1つのスレッドを処理した場合の最大スループットは平均で 50 TPS になります。

水平スケーリングには、全体的なキュースループットを高めるために、メッセージのプロデューサー (SendMessage リクエストを生成) とコンシューマー (ReceiveMessage リクエストと DeleteMessage リクエストを生成) の数を増やすことが必要です。水平スケーリングを行うには 3 つの方法があります。

- クライアントあたりのスレッドの数を増やす
- クライアントを追加する
- クライアントあたりのスレッドの数を増やし、クライアントを追加する

クライアントを追加すると、基本的にはキューのスループットが直線的に向上します。たとえば、クライアントの数を 2 倍にした場合、スループットも 2 倍になります。

> **Note**
>
> 水平スケーリングを行うときは、Amazon SQSクライアントがリクエストの送信とレスポンスの受信する同時メッセージプロデューサーとコンシューマーの数を十分にサポートできる接続数またはスレッドがあることを確認してください。例えば、デフォルトでは、クラスのインスタンスは AWS SDK for Java AmazonSQSClient Amazon SQS への接続を最大 50 個維持します。追加の同時プロデューサーおよびコンシューマーを作成するには、AmazonSQSClientBuilder オブジェクトにおいて許容されるプロデューサーおよびコンシューマースレッドの最大数を調整する必要があります。例えば:
>
> ```
> final AmazonSQS sqsClient = AmazonSQSClientBuilder.standard()
>         .withClientConfiguration(new ClientConfiguration()
>                 .withMaxConnections(producerCount + consumerCount))
>         .build();
> ```
>
> AmazonSQSAsyncClient の場合、十分なスレッドがあることも確認する必要があります。この例では、Java v. 1.x でのみ動作します。

アクションバッチ処理

バッチ処理では、サービスへの各ラウンドトリップでより多くの処理が実行されます (たとえば、1 回の SendMessageBatch リクエストで複数のメッセージを送信する場合など)。Amazon SQS バッチ アクションは、SendMessageBatch、DeleteMessageBatch、および ChangeMessageVisibilityBatch です。プロデューサーやコンシューマーを変更することなくバッチ処理を活用するには、Amazon SQSバファリング非同期クライアント を使用します。

> **Note**
>
> ReceiveMessage は、一度に 10 件のメッセージを処理できるため、ReceiveMessageBatch アクションはありません。

バッチ処理では、1 件のメッセージのレイテンシー全体が受け入れられるのではなく、1 回のバッチリクエストの複数のメッセージにまたがるバッチアクションのレイテンシーが分散されます (たとえば、SendMessage リクエストなど)。各ラウンドトリップがより多くの処理を実行するため、バッチリクエストがスレッドと接続をより効率的に使用するようになり、スループットが向上します。

マッチ処理と水平スケーリングを組み合わせて、個々のメッセージリクエストより少ないスレッド、接続、リクエストで一定のスループットを実現できます。バッチ処理されたAmazon SQSアクションを使用して、最大 10 通のメッセージを一度に送信、受信、または削除できます。Amazon SQSではリクエスト単位で課金されるため、バッチ処理はコストを大幅に削減できます。

バッチ処理により、アプリケーションがいくらか複雑になる可能性はあります (たとえば、アプリケーションはメッセージを送信前に累積する必要があります。または、レスポンスを長時間待機する必要が生じることもときどきあります)。しかし、それでもバッチ処理は次の場合に効果的です:

- アプリケーションが短い時間で多くのメッセージを生成するため、遅延が大幅に長くなることはない。
- 一般的なメッセージプロデューサーが自身でコントロールしていないイベントに応答してメッセージを送信する必要があるのと異なり、メッセージコンシューマーは自身の判断でキューからメッセージを取得する。

> **Important**
>
> バッチ内の個々のメッセージが失敗しても、バッチリクエストは成功することがあります。バッチリクエストの後、必ず個々のメッセージのエラーがないか確認し、必要に応じてアクションを再試行してください。

1 回のオペレーションおよびバッチリクエストでの Java の使用例

前提条件

aws-java-sdk-sqs.jarパッケージ、aws-java-sdk-ec2.jar パッケージおよび commons-logging.jar パッケージを Java ビルドクラスパスに追加します。以下の例は、Maven プロジェクトの pom.xml ファイルにあるこれらの依存関係を示しています。

```
<dependencies>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-java-sdk-sqs</artifactId>
        <version>LATEST</version>
    </dependency>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-java-sdk-ec2</artifactId>
        <version>LATEST</version>
    </dependency>
    <dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>LATEST</version>
    </dependency>
</dependencies>
```

SimpleProducerConsumer.java

次の Java コード例では、簡単なプロデューサー-コンシューマーパターンが実行されています。メインスレッドでは、指定された時間に 1 KB のメッセージを処理するプロデューサーおよびコンシューマースレッドが多数発生します。この例には、単一オペレーションリクエストを生成するプロデューサーおよびコンシューマーと、バッチ処理リクエストを生成するプロデューサーおよびコンシューマーが含まれています。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

import com.amazonaws.AmazonClientException;
import com.amazonaws.ClientConfiguration;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.*;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import java.math.BigInteger;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * Start a specified number of producer and consumer threads, and produce-consume
 * for the least of the specified duration and 1 hour. Some messages can be left
 * in the queue because producers and consumers might not be in exact balance.
 */
public class SimpleProducerConsumer {

    // The maximum runtime of the program.
    private final static int MAX_RUNTIME_MINUTES = 60;
    private final static Log log = LogFactory.getLog(SimpleProducerConsumer.class);
```

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

        final Scanner input = new Scanner(System.in);

        System.out.print("Enter the queue name: ");
        final String queueName = input.nextLine();

        System.out.print("Enter the number of producers: ");
        final int producerCount = input.nextInt();

        System.out.print("Enter the number of consumers: ");
        final int consumerCount = input.nextInt();

        System.out.print("Enter the number of messages per batch: ");
        final int batchSize = input.nextInt();

        System.out.print("Enter the message size in bytes: ");
        final int messageSizeByte = input.nextInt();

        System.out.print("Enter the run time in minutes: ");
        final int runTimeMinutes = input.nextInt();

        /*
         * Create a new instance of the builder with all defaults (credentials
         * and region) set automatically. For more information, see Creating
         * Service Clients in the AWS SDK for Java Developer Guide.
         */
        final ClientConfiguration clientConfiguration = new ClientConfiguration()
                .withMaxConnections(producerCount + consumerCount);

        final AmazonSQS sqsClient = AmazonSQSClientBuilder.standard()
                .withClientConfiguration(clientConfiguration)
                .build();

        final String queueUrl = sqsClient
                .getQueueUrl(new GetQueueUrlRequest(queueName)).getQueueUrl();

        // The flag used to stop producer, consumer, and monitor threads.
        final AtomicBoolean stop = new AtomicBoolean(false);

        // Start the producers.
        final AtomicInteger producedCount = new AtomicInteger();
        final Thread[] producers = new Thread[producerCount];
```

```
        for (int i = 0; i < producerCount; i++) {
            if (batchSize == 1) {
                producers[i] = new Producer(sqsClient, queueUrl, messageSizeByte,
                        producedCount, stop);
            } else {
                producers[i] = new BatchProducer(sqsClient, queueUrl, batchSize,
                        messageSizeByte, producedCount,
                        stop);
            }
            producers[i].start();
        }

        // Start the consumers.
        final AtomicInteger consumedCount = new AtomicInteger();
        final Thread[] consumers = new Thread[consumerCount];
        for (int i = 0; i < consumerCount; i++) {
            if (batchSize == 1) {
                consumers[i] = new Consumer(sqsClient, queueUrl, consumedCount,
                        stop);
            } else {
                consumers[i] = new BatchConsumer(sqsClient, queueUrl, batchSize,
                        consumedCount, stop);
            }
            consumers[i].start();
        }

        // Start the monitor thread.
        final Thread monitor = new Monitor(producedCount, consumedCount, stop);
        monitor.start();

        // Wait for the specified amount of time then stop.
        Thread.sleep(TimeUnit.MINUTES.toMillis(Math.min(runTimeMinutes,
                MAX_RUNTIME_MINUTES)));
        stop.set(true);

        // Join all threads.
        for (int i = 0; i < producerCount; i++) {
            producers[i].join();
        }

        for (int i = 0; i < consumerCount; i++) {
            consumers[i].join();
        }
```

```
        monitor.interrupt();
        monitor.join();
    }

    private static String makeRandomString(int sizeByte) {
        final byte[] bs = new byte[(int) Math.ceil(sizeByte * 5 / 8)];
        new Random().nextBytes(bs);
        bs[0] = (byte) ((bs[0] | 64) & 127);
        return new BigInteger(bs).toString(32);
    }

    /**
     * The producer thread uses {@code SendMessage}
     * to send messages until it is stopped.
     */
    private static class Producer extends Thread {
        final AmazonSQS sqsClient;
        final String queueUrl;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        Producer(AmazonSQS sqsQueueBuffer, String queueUrl, int messageSizeByte,
                 AtomicInteger producedCount, AtomicBoolean stop) {
            this.sqsClient = sqsQueueBuffer;
            this.queueUrl = queueUrl;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /*
         * The producedCount object tracks the number of messages produced by
         * all producer threads. If there is an error, the program exits the
         * run() method.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    sqsClient.sendMessage(new SendMessageRequest(queueUrl,
                            theMessage));
                    producedCount.incrementAndGet();
                }
            } catch (AmazonClientException e) {
```

```
                /*
                 * By default, AmazonSQSClient retries calls 3 times before
                 * failing. If this unlikely condition occurs, stop.
                 */
                log.error("Producer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * The producer thread uses {@code SendMessageBatch}
     * to send messages until it is stopped.
     */
    private static class BatchProducer extends Thread {
        final AmazonSQS sqsClient;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        BatchProducer(AmazonSQS sqsQueueBuffer, String queueUrl, int batchSize,
                      int messageSizeByte, AtomicInteger producedCount,
                      AtomicBoolean stop) {
            this.sqsClient = sqsQueueBuffer;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        public void run() {
            try {
                while (!stop.get()) {
                    final SendMessageBatchRequest batchRequest =
                            new SendMessageBatchRequest().withQueueUrl(queueUrl);

                    final List<SendMessageBatchRequestEntry> entries =
                            new ArrayList<SendMessageBatchRequestEntry>();
                    for (int i = 0; i < batchSize; i++)
                        entries.add(new SendMessageBatchRequestEntry()
                                .withId(Integer.toString(i))
```

```
                            .withMessageBody(theMessage));
                batchRequest.setEntries(entries);

                final SendMessageBatchResult batchResult =
                        sqsClient.sendMessageBatch(batchRequest);
                producedCount.addAndGet(batchResult.getSuccessful().size());

                /*
                 * Because SendMessageBatch can return successfully, but
                 * individual batch items fail, retry the failed batch items.
                 */
                if (!batchResult.getFailed().isEmpty()) {
                    log.warn("Producer: retrying sending "
                            + batchResult.getFailed().size() + " messages");
                    for (int i = 0, n = batchResult.getFailed().size();
                         i < n; i++) {
                        sqsClient.sendMessage(new
                                SendMessageRequest(queueUrl, theMessage));
                        producedCount.incrementAndGet();
                    }
                }
            }
        } catch (AmazonClientException e) {
            /*
             * By default, AmazonSQSClient retries calls 3 times before
             * failing. If this unlikely condition occurs, stop.
             */
            log.error("BatchProducer: " + e.getMessage());
            System.exit(1);
        }
    }
}

/**
 * The consumer thread uses {@code ReceiveMessage} and {@code DeleteMessage}
 * to consume messages until it is stopped.
 */
private static class Consumer extends Thread {
    final AmazonSQS sqsClient;
    final String queueUrl;
    final AtomicInteger consumedCount;
    final AtomicBoolean stop;

    Consumer(AmazonSQS sqsClient, String queueUrl, AtomicInteger consumedCount,
```

```
                AtomicBoolean stop) {
            this.sqsClient = sqsClient;
            this.queueUrl = queueUrl;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /*
         * Each consumer thread receives and deletes messages until the main
         * thread stops the consumer thread. The consumedCount object tracks the
         * number of messages that are consumed by all consumer threads, and the
         * count is logged periodically.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    try {
                        final ReceiveMessageResult result = sqsClient
                                .receiveMessage(new
                                        ReceiveMessageRequest(queueUrl));

                        if (!result.getMessages().isEmpty()) {
                            final Message m = result.getMessages().get(0);
                            sqsClient.deleteMessage(new
                                    DeleteMessageRequest(queueUrl,
                                    m.getReceiptHandle()));
                            consumedCount.incrementAndGet();
                        }
                    } catch (AmazonClientException e) {
                        log.error(e.getMessage());
                    }
                }
            } catch (AmazonClientException e) {
                /*
                 * By default, AmazonSQSClient retries calls 3 times before
                 * failing. If this unlikely condition occurs, stop.
                 */
                log.error("Consumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
```

```
     * The consumer thread uses {@code ReceiveMessage} and {@code
     * DeleteMessageBatch} to consume messages until it is stopped.
     */
    private static class BatchConsumer extends Thread {
        final AmazonSQS sqsClient;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger consumedCount;
        final AtomicBoolean stop;

        BatchConsumer(AmazonSQS sqsClient, String queueUrl, int batchSize,
                      AtomicInteger consumedCount, AtomicBoolean stop) {
            this.sqsClient = sqsClient;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        public void run() {
            try {
                while (!stop.get()) {
                    final ReceiveMessageResult result = sqsClient
                            .receiveMessage(new ReceiveMessageRequest(queueUrl)
                                    .withMaxNumberOfMessages(batchSize));

                    if (!result.getMessages().isEmpty()) {
                        final List<Message> messages = result.getMessages();
                        final DeleteMessageBatchRequest batchRequest =
                                new DeleteMessageBatchRequest()
                                        .withQueueUrl(queueUrl);

                        final List<DeleteMessageBatchRequestEntry> entries =
                                new ArrayList<DeleteMessageBatchRequestEntry>();
                        for (int i = 0, n = messages.size(); i < n; i++)
                            entries.add(new DeleteMessageBatchRequestEntry()
                                    .withId(Integer.toString(i))
                                    .withReceiptHandle(messages.get(i)
                                            .getReceiptHandle()));
                        batchRequest.setEntries(entries);

                        final DeleteMessageBatchResult batchResult = sqsClient
                                .deleteMessageBatch(batchRequest);
                        consumedCount.addAndGet(batchResult.getSuccessful().size());
```

```
                    /*
                     * Because DeleteMessageBatch can return successfully,
                     * but individual batch items fail, retry the failed
                     * batch items.
                     */
                    if (!batchResult.getFailed().isEmpty()) {
                        final int n = batchResult.getFailed().size();
                        log.warn("Producer: retrying deleting " + n
                                + " messages");
                        for (BatchResultErrorEntry e : batchResult
                                .getFailed()) {

                            sqsClient.deleteMessage(
                                    new DeleteMessageRequest(queueUrl,
                                            messages.get(Integer
                                                    .parseInt(e.getId()))
                                                    .getReceiptHandle()));

                            consumedCount.incrementAndGet();
                        }
                    }
                }
            } catch (AmazonClientException e) {
                /*
                 * By default, AmazonSQSClient retries calls 3 times before
                 * failing. If this unlikely condition occurs, stop.
                 */
                log.error("BatchConsumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * This thread prints every second the number of messages produced and
     * consumed so far.
     */
    private static class Monitor extends Thread {
        private final AtomicInteger producedCount;
        private final AtomicInteger consumedCount;
        private final AtomicBoolean stop;
```

```
        Monitor(AtomicInteger producedCount, AtomicInteger consumedCount,
                AtomicBoolean stop) {
            this.producedCount = producedCount;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        public void run() {
            try {
                while (!stop.get()) {
                    Thread.sleep(1000);
                    log.info("produced messages = " + producedCount.get()
                            + ", consumed messages = " + consumedCount.get());
                }
            } catch (InterruptedException e) {
                // Allow the thread to exit.
            }
        }
    }
}
```

サンプル実行からのボリュームメトリクスのモニタリング

Amazon SQSは、メッセージの送信、受信、削除のボリュームメトリクスを自動的に生成します。これらのメトリクスなどには、キューのモニタリングタブまたはCloudWatch コンソール からアクセスできます。

> **Note**
>
> メトリクスを、キューが開始してから参照可能になるまで最大 15 分かかる場合があります。

JMSとAmazon SQSを使用した作業

Amazon SQS Java Message Service (JMS)ライブラリ はAmazon SQS 用のJava Message Service (JMS)インターフェイスで、既にJMSを使用しているアプリケーションでAmazon SQS を利用できます。このインターフェイスにより、最小限のコードの変更で、 Amazon SQS をJMSプロバイダーとして使用できます。 AWS SDK for Javaと一緒に Amazon SQS Java Messaging Library を使用すると、JMS 接続およびセッション、Amazon SQS キューとの間でメッセージを送受信するプロデューサーおよびコンシューマーを作成することができます。

ライブラリは、JMS point-to-point 1.1 仕様 に従って、キュー (JMS モデル) へのメッセージの送受信をサポートします。ライブラリは、テキスト、バイト、または Amazon SQSキューへのオブジェクトメッセージの同期的な送信をサポートしています。また、オブジェクトの同期的または非同期的な受信もサポートしています。

JMS1.1仕様をサポートするAmazon SQS Java Messaging Library の機能の詳細については、「Amazon SQS でサポートされている JMS 1.1 の実装 および Amazon SQS に関するよくある質問」を参照してください。

トピック

JMS と Amazon SQS を使用するための前提条件

始めるには以下の前提条件を満たす必要があります:

- SDK for Java

  プロジェクトにSDK for Javaを含めるには以下の2つの方法があります:

  - SDK for Java をダウンロードしてインストールします。
  - Mavenを使用してAmazon SQS Java メッセージングライブラリを入手します

> **Note**
>
> SDK for Javaは依存関係として含められます。
> SDK for JavaとJava用Amazon SQS 拡張クライアントライブラリには、J2SE Devopment Kit 8.0 以降が必要です。

SDK for Java のダウンロードの詳細については、「 SDK for Java」を参照してください。

- Amazon SQS Java メッセージングライブラリ

  Mavenを使用しない場合は、`amazon-sqs-java-messaging-lib.jar`パッケージをJavaクラスパスに追加する必要があります。ライブラリのダウンロードの詳細については、「Amazon SQS Java メッセージングライブラリ」を参照してください。

  > **Note**
  >
  > Amazon SQS Java メッセージングライブラリには、MavenとSpring フレームワークのサポートが含まれます。
  > Maven、Springフレームワーク、およびAmazon SQS Java メッセージングライブラリを使用するサンプルコードについては、「 Amazon SQS 標準キューで JMS を使用するための Java の使用例」を参照してください。

  ```
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>amazon-sqs-java-messaging-lib</artifactId>
    <version>1.0.4</version>
    <type>jar</type>
  </dependency>
  ```

- Amazon SQS キュー

  for AWS Management Console Amazon SQS 、 API、または Amazon SQS Java メッセージングライブラリに含まれているラップされた Amazon SQS クライアントを使用してキューを作成します。 `CreateQueue`

  - Amazon SQS AWS Management Console または`CreateQueue`の APIを使用してキューを作成する方法については、「キューの作成」を参照してください。

- Amazon SQS Java メッセージングライブラリの使用の詳細については、「Amazon SQS Java メッセージングライブラリの使用を開始する」を参照してください。

Amazon SQS Javaメッセージングライブラリの使用を開始する

Amazon SQSでJava Message Service (JMS)の使用を開始するには、このセクションのコード例を使用します。以降のセクションでは、JMS接続およびセッションの作成方法や、メッセージの送受信方法を説明します。

Amazon SQS Javaメッセージングライブラリに含まれるラップされた Amazon SQS クライアントオブジェクトは、Amazon SQS キューが存在するかどうかをチェックします。キューが存在しない場合は、クライアントがキューを作成します。

JMS接続の作成

1. 接続ファクトリを作成し、そのファクトリに対してcreateConnection メソッドを呼び出します。

```
// Create a new connection factory with all defaults (credentials and region) set
 automatically
SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
        new ProviderConfiguration(),
        AmazonSQSClientBuilder.defaultClient()
        );

// Create the connection.
SQSConnection connection = connectionFactory.createConnection();
```

SQSConnectionクラスは javax.jms.Connection を拡張します。JMSのスタンダード接続メソッドと共に、SQSConnectionは、getAmazonSQSClientやgetWrappedAmazonSQSClientなどの追加のメソッドも提供します。どちらのメッソドでも、新しいキューの作成などのJMS仕様には含まれていない管理操作を実行できます。ただし、getWrappedAmazonSQSClientメソッドは現在接続で使用されているAmazon SQSクライアントのラップされたバージョンも提供します。ラッパーはクライアントからのすべての例外を JMSException に変換するので、JMSException を想定する既存のコードでより容易に使用できます。

2. getAmazonSQSClientやgetWrappedAmazonSQSClientから返されるクライアントオブジェクトを使用して、JMS 仕様には含まれていない管理操作 (Amazon SQS キューの作成など) を実行できます。

   既存のコードで JMS 例外を想定している場合は、getWrappedAmazonSQSClient を使用する必要があります:

   - getWrappedAmazonSQSClientを使用する場合、返されるクライアントオブジェクトはすべての例外をJMS例外に変換します。
   - getAmazonSQSClientを使用する場合、例外はすべてAmazon SQS例外になります。

Amazon SQSキューを作成する

ラップされたクライアントオブジェクトは、 Amazon SQSキューが存在するかどうかを確認します。

キューが存在しない場合は、クライアントがキューを作成します。キューが存在する場合、関数からは何も返されません。詳細については、TextMessageSender.java にある「必要に応じてキューを作成する」セクションを参照してください。

標準キューを作成するには

```
// Get the wrapped client
AmazonSQSMessagingClientWrapper client = connection.getWrappedAmazonSQSClient();

// Create an SQS queue named MyQueue, if it doesn't already exist
if (!client.queueExists("MyQueue")) {
    client.createQueue("MyQueue");
}
```

FIFOキューを作成するには

```
// Get the wrapped client
AmazonSQSMessagingClientWrapper client = connection.getWrappedAmazonSQSClient();

// Create an Amazon SQS FIFO queue named MyQueue.fifo, if it doesn't already exist
if (!client.queueExists("MyQueue.fifo")) {
    Map<String, String> attributes = new HashMap<String, String>();
    attributes.put("FifoQueue", "true");
```

```
    attributes.put("ContentBasedDeduplication", "true");
    client.createQueue(new
 CreateQueueRequest().withQueueName("MyQueue.fifo").withAttributes(attributes));
}
```

> **Note**
>
> FIFO キュー名は.fifoのサフィックスで終わる必要があります。
> ContentBasedDeduplication属性の詳細については、「Amazon SQS での 1 回限りの処理」を参照してください。

同期的なメッセージの送信

1. 接続と基になる Amazon SQS キューの準備ができたら、 AUTO_ACKNOWLEDGEモードで非トランザクションJMSセッションを作成します。

```
// Create the nontransacted session with AUTO_ACKNOWLEDGE mode
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
```

2. テキストメッセージをキューに送信するには、JMS キュー ID とメッセージプロデューサーを作成します。

```
// Create a queue identity and specify the queue name to the session
Queue queue = session.createQueue("MyQueue");

// Create a producer for the 'MyQueue'
MessageProducer producer = session.createProducer(queue);
```

3. テキストメッセージを作成し、キューに送信します。

   - メッセージをスタンダード キューに送信するために、追加のパラメーターを設定する必要はありません。

```
// Create the text message
TextMessage message = session.createTextMessage("Hello World!");

// Send the message
producer.send(message);
System.out.println("JMS Message " + message.getJMSMessageID());
```

- メッセージをFIFOキューに送信するには、メッセージのグループ IDを設定する必要があります。メッセージ重複排除IDを設定することもできます。詳細については、「Amazon SQS の主要な用語」を参照してください。

```
// Create the text message
TextMessage message = session.createTextMessage("Hello World!");

// Set the message group ID
message.setStringProperty("JMSXGroupID", "Default");

// You can also set a custom message deduplication ID
// message.setStringProperty("JMS_SQS_DeduplicationId", "hello");
// Here, it's not needed because content-based deduplication is enabled for the
 queue

// Send the message
producer.send(message);
System.out.println("JMS Message " + message.getJMSMessageID());
System.out.println("JMS Message Sequence Number " +
 message.getStringProperty("JMS_SQS_SequenceNumber"));
```

同期的なメッセージの受信

1. メッセージを受信するには、同じキューにコンシューマーを作成し、startメソッドを呼び出します。

   接続でのstartメソッドはいつでも呼び出すことができます。ただし、コンシューマーは呼び出されるまでメッセージの受信を開始しません。

```
// Create a consumer for the 'MyQueue'
MessageConsumer consumer = session.createConsumer(queue);
// Start receiving incoming messages
connection.start();
```

2. コンシューマーで、タイムアウトを1秒に設定してreceiveメソッドを呼び出し、受信メッセージの内容を出力します。

   - スタンダードキューからメッセージを受信したら、メッセージの内容にアクセスできます。

```
// Receive a message from 'MyQueue' and wait up to 1 second
```

```
Message receivedMessage = consumer.receive(1000);

// Cast the received message as TextMessage and display the text
if (receivedMessage != null) {
    System.out.println("Received: " + ((TextMessage) receivedMessage).getText());
}
```

- FIFOキューからメッセージを受信したら、メッセージの内容や、その他のFIFO 固有のメッセージ属性 (メッセージグループ ID、メッセージ重複排除 ID、シーケンス番号など) にアクセスできます。詳細については、「Amazon SQS の主要な用語」を参照してください。

```
// Receive a message from 'MyQueue' and wait up to 1 second
Message receivedMessage = consumer.receive(1000);

// Cast the received message as TextMessage and display the text
if (receivedMessage != null) {
    System.out.println("Received: " + ((TextMessage) receivedMessage).getText());
    System.out.println("Group id: " +
 receivedMessage.getStringProperty("JMSXGroupID"));
    System.out.println("Message deduplication id: " +
 receivedMessage.getStringProperty("JMS_SQS_DeduplicationId"));
    System.out.println("Message sequence number: " +
 receivedMessage.getStringProperty("JMS_SQS_SequenceNumber"));
}
```

3. 接続とセッションを閉じます。

```
// Close the connection (and the session).
connection.close();
```

出力は次の例のようになります:

```
JMS Message ID:8example-588b-44e5-bbcf-d816example2
Received: Hello World!
```

> **Note**
>
> Spring Framework を使用してこれらのオブジェクトを初期化できます。
> 追加情報については、
> 「SpringExampleConfiguration.xml」、SpringExample.java、およ

びExampleConfiguration.javaのほか、ExampleCommon.javaおよびAmazon SQS 標準キューで JMS を使用するための Java の使用例 に含まれる他のヘルパークラスを参照してください。

オブジェクトの送受信の完全な例については、TextMessageSender.javaおよびSyncMessageReceiver.javaを参照してください。

非同期的なメッセージの受信

「Amazon SQS Javaメッセージングライブラリの使用を開始する」の例では、メッセージはMyQueueに送信され、同期的に受信されます。

以下の例では、リスナーを介してメッセージを非同期的に受信する方法を示します。

1. MessageListenerインターフェイスを実装します。

```
class MyListener implements MessageListener {

    @Override
    public void onMessage(Message message) {
        try {
            // Cast the received message as TextMessage and print the text to
 screen.
            System.out.println("Received: " + ((TextMessage) message).getText());
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```

   onMessageインターフェイスのMessageListenerメソッドは、メッセージを受信すると呼び出されます。このリスナーの実装内で、メッセージに格納されたテキストが出力されます。

2. コンシューマーで明示的にreceiveメソッドを呼び出す代わりに、コンシューマーのメッセージリスナーをMyListener実装のインスタンスに設定します。メインスレッドは1秒間待機します。

```
// Create a consumer for the 'MyQueue'.
MessageConsumer consumer = session.createConsumer(queue);
```

```
// Instantiate and set the message listener for the consumer.
consumer.setMessageListener(new MyListener());

// Start receiving incoming messages.
connection.start();

// Wait for 1 second. The listener onMessage() method is invoked when a message is
 received.
Thread.sleep(1000);
```

残りの手順は、「Amazon SQS Javaメッセージングライブラリの使用を開始する」の例の手順と同じです。非同期コンシューマーの完全な例については、「AsyncMessageReceiver.java」のAmazon SQS 標準キューで JMS を使用するための Java の使用例を参照してください。

この例の出力は以下の例のようになります:

```
JMS Message ID:8example-588b-44e5-bbcf-d816example2
Received: Hello World!
```

クライアント確認モードの使用

「Amazon SQS Javaメッセージングライブラリの使用を開始する」の例ではAUTO_ACKNOWLEDGEモードを使用しています。この場合、受信したすべてのメッセージは自動的に確認されます (このため、基になるAmazon SQSキューから削除されます)。

1. メッセージが処理された後にメッセージを明示的に確認するには、CLIENT_ACKNOWLEDGEモードでセッションを作成する必要があります。

   ```
   // Create the non-transacted session with CLIENT_ACKNOWLEDGE mode.
   Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
   ```

2. メッセージが受信されたら、メッセージを表示して、その後明示的に確認します。

   ```
   // Cast the received message as TextMessage and print the text to screen. Also
    acknowledge the message.
   if (receivedMessage != null) {
       System.out.println("Received: " + ((TextMessage) receivedMessage).getText());
       receivedMessage.acknowledge();
       System.out.println("Acknowledged: " + message.getJMSMessageID());
   }
   ```

> **Note**
>
> このモードでは、メッセージが確認されると、そのメッセージ以前に受信されたすべてのメッセージも暗黙的に確認されます。たとえば、10通のメッセージが受信され、(受信した順序で)10番目のメッセージのみが確認された場合、前の9通のメッセージもすべて確認されます。

残りの手順は、「Amazon SQS Javaメッセージングライブラリの使用を開始する」の例の手順と同じです。クライアント確認モードの同期コンシューマーの完全な例については、「SyncMessageReceiverClientAcknowledge.java」のAmazon SQS 標準キューで JMS を使用するための Java の使用例を参照してください。

この例の出力は以下の例のようになります:

```
JMS Message ID:4example-aa0e-403f-b6df-5e02example5
Received: Hello World!
Acknowledged: ID:4example-aa0e-403f-b6df-5e02example5
```

順不同確認モードの使用

CLIENT_ACKNOWLEDGEモードを使用すると、明示的に確認されたメッセージの前に受信されたすべてのメッセージが自動的に確認されます。詳細については、「クライアント確認モードの使用」を参照してください。

Amazon SQS Java メッセージングライブラリには別の確認モードも用意されています。UNORDERED_ACKNOWLEDGE モードを使用する場合は、受信した順序に関係なく、すべての受信メッセージを個別かつ明示的にクライアントが確認する必要があります。これには、UNORDERED_ACKNOWLEDGEモードでセッションを作成します。

```
// Create the non-transacted session with UNORDERED_ACKNOWLEDGE mode.
Session session = connection.createSession(false, SQSSession.UNORDERED_ACKNOWLEDGE);
```

残りの手順は、「クライアント確認モードの使用」の例の手順と同じです。UNORDERED_ACKNOWLEDGEモードの同期コンシューマーの完全な例については、SyncMessageReceiverUnorderedAcknowledge.javaを参照してください。

この例の出力は以下のようになります:

```
JMS Message ID:dexample-73ad-4adb-bc6c-4357example7
Received: Hello World!
Acknowledged: ID:dexample-73ad-4adb-bc6c-4357example7
```

Java Message Service を他の Amazon SQS クライアントで使用する

AWS SDK で Amazon SQS Java Message Service (JMS) クライアントを使用すると、Amazon SQS メッセージサイズは 256 KB に制限されます。ただし、任意のAmazon SQSクライアントを使用してJMSプロバイダを作成することができます。たとえば、Java用の Amazon SQS 拡張クライアントライブラリ とJMSクライアントを使用する場合、Amazon S3内のメッセージペイロード (最大2GB) への参照を含むAmazon SQSメッセージを送信することができます。詳細については、「Java と Amazon S3 を使用した大規模な Amazon SQS Amazon S3の管理」を参照してください。

次のJavaコード例では、拡張クライアントライブラリ のJMSプロバイダを作成しています:

```
AmazonS3 s3 = new AmazonS3Client(credentials);
Region s3Region = Region.getRegion(Regions.US_WEST_2);
s3.setRegion(s3Region);

// Set the Amazon S3 bucket name, and set a lifecycle rule on the bucket to
// permanently delete objects a certain number of days after each object's creation
 date.
// Next, create the bucket, and enable message objects to be stored in the bucket.
BucketLifecycleConfiguration.Rule expirationRule = new
 BucketLifecycleConfiguration.Rule();
expirationRule.withExpirationInDays(14).withStatus("Enabled");
BucketLifecycleConfiguration lifecycleConfig = new
 BucketLifecycleConfiguration().withRules(expirationRule);

s3.createBucket(s3BucketName);
s3.setBucketLifecycleConfiguration(s3BucketName, lifecycleConfig);
System.out.println("Bucket created and configured.");

// Set the SQS extended client configuration with large payload support enabled.
ExtendedClientConfiguration extendedClientConfig = new ExtendedClientConfiguration()
    .withLargePayloadSupportEnabled(s3, s3BucketName);

AmazonSQS sqsExtended = new AmazonSQSExtendedClient(new AmazonSQSClient(credentials),
 extendedClientConfig);
```

```
Region sqsRegion = Region.getRegion(Regions.US_WEST_2);
sqsExtended.setRegion(sqsRegion);
```

次のJavaコード例は、接続ファクトリを作成しています:

```
// Create the connection factory using the environment variable credential provider.
// Pass the configured Amazon SQS Extended Client to the JMS connection factory.
SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
        new ProviderConfiguration(),
        sqsExtended
        );

// Create the connection.
SQSConnection connection = connectionFactory.createConnection();
```

Amazon SQS 標準キューで JMS を使用するための Java の使用例

次のコード例で、Java Message Service (JMS)を Amazon SQS スタンダードキューと共に使用する方法を示します。FIFO キューの操作方法の詳細については、「FIFOキューを作成するには」、「同期的なメッセージの送信」、および「同期的なメッセージの受信」を参照してください。
(メッセージを同期して受信することは、スタンダードキューでもFIFOキューでも同じです。 ただし、FIFOキューのメッセージには、より多くの属性が含まれます)。

ExampleConfiguration.java

次の Java SDK v 1.x コード例では、デフォルトのキュー名、リージョン、および他の Java の例で使用する認証情報を設定します。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
```

```
 *
 */

public class ExampleConfiguration {
    public static final String DEFAULT_QUEUE_NAME = "SQSJMSClientExampleQueue";

    public static final Region DEFAULT_REGION = Region.getRegion(Regions.US_EAST_2);

    private static String getParameter( String args[], int i ) {
        if( i + 1 >= args.length ) {
            throw new IllegalArgumentException( "Missing parameter for " + args[i] );
        }
        return args[i+1];
    }

    /**
     * Parse the command line and return the resulting config. If the config parsing
 fails
     * print the error and the usage message and then call System.exit
     *
     * @param app the app to use when printing the usage string
     * @param args the command line arguments
     * @return the parsed config
     */
    public static ExampleConfiguration parseConfig(String app, String args[]) {
        try {
            return new ExampleConfiguration(args);
        } catch (IllegalArgumentException e) {
            System.err.println( "ERROR: " + e.getMessage() );
            System.err.println();
            System.err.println( "Usage: " + app + " [--queue <queue>] [--region
 <region>] [--credentials <credentials>] ");
            System.err.println( "  or" );
            System.err.println( "       " + app + " <spring.xml>" );
            System.exit(-1);
            return null;
        }
    }

    private ExampleConfiguration(String args[]) {
        for( int i = 0; i < args.length; ++i ) {
            String arg = args[i];
            if( arg.equals( "--queue" ) ) {
                setQueueName(getParameter(args, i));
```

```
                i++;
            } else if( arg.equals( "--region" ) ) {
                String regionName = getParameter(args, i);
                try {
                    setRegion(Region.getRegion(Regions.fromName(regionName)));
                } catch( IllegalArgumentException e ) {
                    throw new IllegalArgumentException( "Unrecognized region " +
regionName );
                }
                i++;
            } else if( arg.equals( "--credentials" ) ) {
                String credsFile = getParameter(args, i);
                try {
                    setCredentialsProvider( new
PropertiesFileCredentialsProvider(credsFile) );
                } catch (AmazonClientException e) {
                    throw new IllegalArgumentException("Error reading credentials from
" + credsFile, e );
                }
                i++;
            } else {
                throw new IllegalArgumentException("Unrecognized option " + arg);
            }
        }
    }

    private String queueName = DEFAULT_QUEUE_NAME;
    private Region region = DEFAULT_REGION;
    private AWSCredentialsProvider credentialsProvider = new
DefaultAWSCredentialsProviderChain();

    public String getQueueName() {
        return queueName;
    }

    public void setQueueName(String queueName) {
        this.queueName = queueName;
    }

    public Region getRegion() {
        return region;
    }

    public void setRegion(Region region) {
```

```
        this.region = region;
    }

    public AWSCredentialsProvider getCredentialsProvider() {
        return credentialsProvider;
    }

    public void setCredentialsProvider(AWSCredentialsProvider credentialsProvider) {
        // Make sure they're usable first
        credentialsProvider.getCredentials();
        this.credentialsProvider = credentialsProvider;
    }
}
```

TextMessageSender.java

次のJavaコード例では、テキストメッセージプロデューサーを作成しています。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

public class TextMessageSender {
    public static void main(String args[]) throws JMSException {
        ExampleConfiguration config =
 ExampleConfiguration.parseConfig("TextMessageSender", args);

        ExampleCommon.setupLogging();

        // Create the connection factory based on the config
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
```

```
                new ProviderConfiguration(),
                AmazonSQSClientBuilder.standard()
                        .withRegion(config.getRegion().getName())
                        .withCredentials(config.getCredentialsProvider())
                );

        // Create the connection
        SQSConnection connection = connectionFactory.createConnection();

        // Create the queue if needed
        ExampleCommon.ensureQueueExists(connection, config.getQueueName());

        // Create the session
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageProducer producer =
session.createProducer( session.createQueue( config.getQueueName() ) );

        sendMessages(session, producer);

        // Close the connection. This closes the session automatically
        connection.close();
        System.out.println( "Connection closed" );
    }

    private static void sendMessages( Session session, MessageProducer producer ) {
        BufferedReader inputReader = new BufferedReader(
            new InputStreamReader( System.in, Charset.defaultCharset() ) );

        try {
            String input;
            while( true ) {
                System.out.print( "Enter message to send (leave empty to exit): " );
                input = inputReader.readLine();
                if( input == null || input.equals("" ) ) break;

                TextMessage message = session.createTextMessage(input);
                producer.send(message);
                System.out.println( "Send message " + message.getJMSMessageID() );
            }
        } catch (EOFException e) {
            // Just return on EOF
        } catch (IOException e) {
            System.err.println( "Failed reading input: " + e.getMessage() );
        } catch (JMSException e) {
```

```
            System.err.println( "Failed sending message: " + e.getMessage() );
            e.printStackTrace();
        }
    }
}
```

SyncMessageReceiver.java

次のJavaコード例では、同期メッセージコンシューマーを作成しています。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

public class SyncMessageReceiver {
public static void main(String args[]) throws JMSException {
    ExampleConfiguration config =
 ExampleConfiguration.parseConfig("SyncMessageReceiver", args);

    ExampleCommon.setupLogging();

    // Create the connection factory based on the config
    SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
            new ProviderConfiguration(),
            AmazonSQSClientBuilder.standard()
                    .withRegion(config.getRegion().getName())
                    .withCredentials(config.getCredentialsProvider())
            );

    // Create the connection
    SQSConnection connection = connectionFactory.createConnection();
```

```
    // Create the queue if needed
    ExampleCommon.ensureQueueExists(connection, config.getQueueName());

    // Create the session
    Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
    MessageConsumer consumer =
 session.createConsumer( session.createQueue( config.getQueueName() ) );

    connection.start();

    receiveMessages(session, consumer);

    // Close the connection. This closes the session automatically
    connection.close();
    System.out.println( "Connection closed" );
}

private static void receiveMessages( Session session, MessageConsumer consumer ) {
    try {
        while( true ) {
            System.out.println( "Waiting for messages");
            // Wait 1 minute for a message
            Message message = consumer.receive(TimeUnit.MINUTES.toMillis(1));
            if( message == null ) {
                System.out.println( "Shutting down after 1 minute of silence" );
                break;
            }
            ExampleCommon.handleMessage(message);
            message.acknowledge();
            System.out.println( "Acknowledged message " + message.getJMSMessageID() );
        }
    } catch (JMSException e) {
        System.err.println( "Error receiving from SQS: " + e.getMessage() );
        e.printStackTrace();
    }
}
}
```

AsyncMessageReceiver.java

次のJavaコード例では、非同期メッセージコンシューマーを作成しています。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

public class AsyncMessageReceiver {
    public static void main(String args[]) throws JMSException, InterruptedException {
        ExampleConfiguration config =
 ExampleConfiguration.parseConfig("AsyncMessageReceiver", args);

        ExampleCommon.setupLogging();

        // Create the connection factory based on the config
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                AmazonSQSClientBuilder.standard()
                        .withRegion(config.getRegion().getName())
                        .withCredentials(config.getCredentialsProvider())
                );

        // Create the connection
        SQSConnection connection = connectionFactory.createConnection();

        // Create the queue if needed
        ExampleCommon.ensureQueueExists(connection, config.getQueueName());

        // Create the session
        Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
        MessageConsumer consumer =
 session.createConsumer( session.createQueue( config.getQueueName() ) );

        // No messages are processed until this is called
```

```
        connection.start();

        ReceiverCallback callback = new ReceiverCallback();
        consumer.setMessageListener( callback );

        callback.waitForOneMinuteOfSilence();
        System.out.println( "Returning after one minute of silence" );

        // Close the connection. This closes the session automatically
        connection.close();
        System.out.println( "Connection closed" );
    }


    private static class ReceiverCallback implements MessageListener {
        // Used to listen for message silence
        private volatile long timeOfLastMessage = System.nanoTime();

        public void waitForOneMinuteOfSilence() throws InterruptedException {
            for(;;) {
                long timeSinceLastMessage = System.nanoTime() - timeOfLastMessage;
                long remainingTillOneMinuteOfSilence =
                    TimeUnit.MINUTES.toNanos(1) - timeSinceLastMessage;
                if( remainingTillOneMinuteOfSilence < 0 ) {
                    break;
                }
                TimeUnit.NANOSECONDS.sleep(remainingTillOneMinuteOfSilence);
            }
        }


        @Override
        public void onMessage(Message message) {
            try {
                ExampleCommon.handleMessage(message);
                message.acknowledge();
                System.out.println( "Acknowledged message " +
message.getJMSMessageID() );
                timeOfLastMessage = System.nanoTime();
            } catch (JMSException e) {
                System.err.println( "Error processing message: " + e.getMessage() );
                e.printStackTrace();
            }
        }
```

```
    }
}
```

SyncMessageReceiverClient確認.Java

次のJavaコード例では、クライアント確認モードの同期コンシューマーを作成しています。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

/**
 * An example class to demonstrate the behavior of CLIENT_ACKNOWLEDGE mode for received
 messages. This example
 * complements the example given in {@link SyncMessageReceiverUnorderedAcknowledge} for
 UNORDERED_ACKNOWLEDGE mode.
 *
 * First, a session, a message producer, and a message consumer are created. Then, two
 messages are sent. Next, two messages
 * are received but only the second one is acknowledged. After waiting for the
 visibility time out period, an attempt to
 * receive another message is made. It's shown that no message is returned for this
 attempt since in CLIENT_ACKNOWLEDGE mode,
 * as expected, all the messages prior to the acknowledged messages are also
 acknowledged.
 *
 * This ISN'T the behavior for UNORDERED_ACKNOWLEDGE mode. Please see {@link
 SyncMessageReceiverUnorderedAcknowledge}
 * for an example.
 */
public class SyncMessageReceiverClientAcknowledge {
```

```
   // Visibility time-out for the queue. It must match to the one set for the queue
for this example to work.
   private static final long TIME_OUT_SECONDS = 1;

   public static void main(String args[]) throws JMSException, InterruptedException {
       // Create the configuration for the example
       ExampleConfiguration config =
ExampleConfiguration.parseConfig("SyncMessageReceiverClientAcknowledge", args);

       // Setup logging for the example
       ExampleCommon.setupLogging();

       // Create the connection factory based on the config
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               AmazonSQSClientBuilder.standard()
                       .withRegion(config.getRegion().getName())
                       .withCredentials(config.getCredentialsProvider())
               );

       // Create the connection
       SQSConnection connection = connectionFactory.createConnection();

       // Create the queue if needed
       ExampleCommon.ensureQueueExists(connection, config.getQueueName());

       // Create the session  with client acknowledge mode
       Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);

       // Create the producer and consume
       MessageProducer producer =
session.createProducer(session.createQueue(config.getQueueName()));
       MessageConsumer consumer =
session.createConsumer(session.createQueue(config.getQueueName()));

       // Open the connection
       connection.start();

       // Send two text messages
       sendMessage(producer, session, "Message 1");
       sendMessage(producer, session, "Message 2");

       // Receive a message and don't acknowledge it
```

```
        receiveMessage(consumer, false);

        // Receive another message and acknowledge it
        receiveMessage(consumer, true);

        // Wait for the visibility time out, so that unacknowledged messages reappear
in the queue
        System.out.println("Waiting for visibility timeout...");
        Thread.sleep(TimeUnit.SECONDS.toMillis(TIME_OUT_SECONDS));

        // Attempt to receive another message and acknowledge it. This results in
receiving no messages since
        // we have acknowledged the second message. Although we didn't explicitly
acknowledge the first message,
        // in the CLIENT_ACKNOWLEDGE mode, all the messages received prior to the
explicitly acknowledged message
        // are also acknowledged. Therefore, we have implicitly acknowledged the first
message.
        receiveMessage(consumer, true);

        // Close the connection. This closes the session automatically
        connection.close();
        System.out.println("Connection closed.");
    }

    /**
     * Sends a message through the producer.
     *
     * @param producer Message producer
     * @param session Session
     * @param messageText Text for the message to be sent
     * @throws JMSException
     */
    private static void sendMessage(MessageProducer producer, Session session, String
messageText) throws JMSException {
        // Create a text message and send it
        producer.send(session.createTextMessage(messageText));
    }

    /**
     * Receives a message through the consumer synchronously with the default timeout
(TIME_OUT_SECONDS).
     * If a message is received, the message is printed. If no message is received,
"Queue is empty!" is
```

```
     * printed.
     *
     * @param consumer Message consumer
     * @param acknowledge If true and a message is received, the received message is
 acknowledged.
     * @throws JMSException
     */
    private static void receiveMessage(MessageConsumer consumer, boolean acknowledge)
 throws JMSException {
        // Receive a message
        Message message =
 consumer.receive(TimeUnit.SECONDS.toMillis(TIME_OUT_SECONDS));

        if (message == null) {
            System.out.println("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object and
 print the text
            System.out.println("Received: " + ((TextMessage) message).getText());

            // Acknowledge the message if asked
            if (acknowledge) message.acknowledge();
        }
    }
}
```

SyncMessageReceiverUnordered確認.Java

次のJavaコード例では、順不同確認モードの同期コンシューマーを作成しています。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
```

```
 *
 */

/**
 * An example class to demonstrate the behavior of UNORDERED_ACKNOWLEDGE mode for
 received messages. This example
 * complements the example given in {@link SyncMessageReceiverClientAcknowledge} for
 CLIENT_ACKNOWLEDGE mode.
 *
 * First, a session, a message producer, and a message consumer are created. Then, two
 messages are sent. Next, two messages
 * are received but only the second one is acknowledged. After waiting for the
 visibility time out period, an attempt to
 * receive another message is made. It's shown that the first message received in the
 prior attempt is returned again
 * for the second attempt. In UNORDERED_ACKNOWLEDGE mode, all the messages must be
 explicitly acknowledged no matter what
 * the order they're received.
 *
 * This ISN'T the behavior for CLIENT_ACKNOWLEDGE mode. Please see {@link
 SyncMessageReceiverClientAcknowledge}
 * for an example.
 */
public class SyncMessageReceiverUnorderedAcknowledge {

    // Visibility time-out for the queue. It must match to the one set for the queue
 for this example to work.
    private static final long TIME_OUT_SECONDS = 1;

    public static void main(String args[]) throws JMSException, InterruptedException {
        // Create the configuration for the example
        ExampleConfiguration config =
 ExampleConfiguration.parseConfig("SyncMessageReceiverUnorderedAcknowledge", args);

        // Setup logging for the example
        ExampleCommon.setupLogging();

        // Create the connection factory based on the config
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                AmazonSQSClientBuilder.standard()
                        .withRegion(config.getRegion().getName())
                        .withCredentials(config.getCredentialsProvider())
                );
```

```
        // Create the connection
        SQSConnection connection = connectionFactory.createConnection();

        // Create the queue if needed
        ExampleCommon.ensureQueueExists(connection, config.getQueueName());

        // Create the session  with unordered acknowledge mode
        Session session = connection.createSession(false,
SQSSession.UNORDERED_ACKNOWLEDGE);

        // Create the producer and consume
        MessageProducer producer =
session.createProducer(session.createQueue(config.getQueueName()));
        MessageConsumer consumer =
session.createConsumer(session.createQueue(config.getQueueName()));

        // Open the connection
        connection.start();

        // Send two text messages
        sendMessage(producer, session, "Message 1");
        sendMessage(producer, session, "Message 2");

        // Receive a message and don't acknowledge it
        receiveMessage(consumer, false);

        // Receive another message and acknowledge it
        receiveMessage(consumer, true);

        // Wait for the visibility time out, so that unacknowledged messages reappear
in the queue
        System.out.println("Waiting for visibility timeout...");
        Thread.sleep(TimeUnit.SECONDS.toMillis(TIME_OUT_SECONDS));

        // Attempt to receive another message and acknowledge it. This results in
receiving the first message since
        // we have acknowledged only the second message. In the UNORDERED_ACKNOWLEDGE
mode, all the messages must
        // be explicitly acknowledged.
        receiveMessage(consumer, true);

        // Close the connection. This closes the session automatically
        connection.close();
```

```
        System.out.println("Connection closed.");
    }

    /**
     * Sends a message through the producer.
     *
     * @param producer Message producer
     * @param session Session
     * @param messageText Text for the message to be sent
     * @throws JMSException
     */
    private static void sendMessage(MessageProducer producer, Session session, String
messageText) throws JMSException {
        // Create a text message and send it
        producer.send(session.createTextMessage(messageText));
    }

    /**
     * Receives a message through the consumer synchronously with the default timeout
(TIME_OUT_SECONDS).
     * If a message is received, the message is printed. If no message is received,
"Queue is empty!" is
     * printed.
     *
     * @param consumer Message consumer
     * @param acknowledge If true and a message is received, the received message is
acknowledged.
     * @throws JMSException
     */
    private static void receiveMessage(MessageConsumer consumer, boolean acknowledge)
throws JMSException {
        // Receive a message
        Message message =
consumer.receive(TimeUnit.SECONDS.toMillis(TIME_OUT_SECONDS));

        if (message == null) {
            System.out.println("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object and
print the text
            System.out.println("Received: " + ((TextMessage) message).getText());

            // Acknowledge the message if asked
            if (acknowledge) message.acknowledge();
```

```
        }
    }
}
```

SpringExampleConfiguration.xml

次のXMLコード例は、SpringExample.javaのBean構成ファイルです。

```
<!--
    Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.

    Licensed under the Apache License, Version 2.0 (the "License").
    You may not use this file except in compliance with the License.
    A copy of the License is located at

    https://aws.amazon.com/apache2.0

    or in the "license" file accompanying this file. This file is distributed
    on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
    express or implied. See the License for the specific language governing
    permissions and limitations under the License.
-->

<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:util="http://www.springframework.org/schema/util"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/
schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/util http://www.springframework.org/
schema/util/spring-util-3.0.xsd
    ">

    <bean id="CredentialsProviderBean"
 class="com.amazonaws.auth.DefaultAWSCredentialsProviderChain"/>

    <bean id="ClientBuilder" class="com.amazonaws.services.sqs.AmazonSQSClientBuilder"
 factory-method="standard">
        <property name="region" value="us-east-2"/>
        <property name="credentials" ref="CredentialsProviderBean"/>
```

```
    </bean>

    <bean id="ProviderConfiguration"
 class="com.amazon.sqs.javamessaging.ProviderConfiguration">
        <property name="numberOfMessagesToPrefetch" value="5"/>
    </bean>

    <bean id="ConnectionFactory"
 class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
        <constructor-arg ref="ProviderConfiguration" />
        <constructor-arg ref="ClientBuilder" />
    </bean>

    <bean id="Connection" class="javax.jms.Connection"
        factory-bean="ConnectionFactory"
        factory-method="createConnection"
        init-method="start"
        destroy-method="close" />

    <bean id="QueueName" class="java.lang.String">
        <constructor-arg value="SQSJMSClientExampleQueue"/>
    </bean>
</beans>
```

SpringExample.java

次のJavaコード例では、Bean構成ファイルを使用してオブジェクトを初期化しています。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */
```

```
public class SpringExample {
    public static void main(String args[]) throws JMSException {
        if( args.length != 1 || !args[0].endsWith(".xml")) {
            System.err.println( "Usage: " + SpringExample.class.getName() + " <spring
 config.xml>" );
            System.exit(1);
        }

        File springFile = new File( args[0] );
        if( !springFile.exists() || !springFile.canRead() ) {
            System.err.println( "File " + args[0] + " doesn't exist or isn't
 readable.");
            System.exit(2);
        }

        ExampleCommon.setupLogging();

        FileSystemXmlApplicationContext context =
            new FileSystemXmlApplicationContext( "file://" +
 springFile.getAbsolutePath() );

        Connection connection;
        try {
            connection = context.getBean(Connection.class);
        } catch( NoSuchBeanDefinitionException e ) {
            System.err.println( "Can't find the JMS connection to use: " +
 e.getMessage() );
            System.exit(3);
            return;
        }

        String queueName;
        try {
            queueName = context.getBean("QueueName", String.class);
        } catch( NoSuchBeanDefinitionException e ) {
            System.err.println( "Can't find the name of the queue to use: " +
 e.getMessage() );
            System.exit(3);
            return;
        }

        if( connection instanceof SQSConnection ) {
            ExampleCommon.ensureQueueExists( (SQSConnection) connection, queueName );
```

```
        }

        // Create the session
        Session session = connection.createSession(false, Session.CLIENT_ACKNOWLEDGE);
        MessageConsumer consumer =
 session.createConsumer( session.createQueue( queueName) );

        receiveMessages(session, consumer);

        // The context can be setup to close the connection for us
        context.close();
        System.out.println( "Context closed" );
    }

    private static void receiveMessages( Session session, MessageConsumer consumer ) {
        try {
            while( true ) {
                System.out.println( "Waiting for messages");
                // Wait 1 minute for a message
                Message message = consumer.receive(TimeUnit.MINUTES.toMillis(1));
                if( message == null ) {
                    System.out.println( "Shutting down after 1 minute of silence" );
                    break;
                }
                ExampleCommon.handleMessage(message);
                message.acknowledge();
                System.out.println( "Acknowledged message" );
            }
        } catch (JMSException e) {
            System.err.println( "Error receiving from SQS: " + e.getMessage() );
            e.printStackTrace();
        }
    }
}
```

ExampleCommon.java

次のJavaコード例では、Amazon SQSキューが存在するかどうかを確認し、存在しない場合はキューを作成します。このコード例にはログ記録コードの例も含まれています。

```
/*
 * Copyright 2010-2024 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
```

```
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

public class ExampleCommon {
    /**
     * A utility function to check the queue exists and create it if needed. For most
     * use cases this is usually done by an administrator before the application is
 run.
     */
    public static void ensureQueueExists(SQSConnection connection, String queueName)
 throws JMSException {
        AmazonSQSMessagingClientWrapper client =
 connection.getWrappedAmazonSQSClient();

        /**
         * In most cases, you can do this with just a createQueue call, but
 GetQueueUrl
         * (called by queueExists) is a faster operation for the common case where the
 queue
         * already exists. Also many users and roles have permission to call
 GetQueueUrl
         * but don't have permission to call CreateQueue.
         */
        if( !client.queueExists(queueName) ) {
            client.createQueue( queueName );
        }
    }

    public static void setupLogging() {
        // Setup logging
        BasicConfigurator.configure();
        Logger.getRootLogger().setLevel(Level.WARN);
    }
```

```
    public static void handleMessage(Message message) throws JMSException {
        System.out.println( "Got message " + message.getJMSMessageID() );
        System.out.println( "Content: ");
        if( message instanceof TextMessage ) {
            TextMessage txtMessage = ( TextMessage ) message;
            System.out.println( "\t" + txtMessage.getText() );
        } else if( message instanceof BytesMessage ){
            BytesMessage byteMessage = ( BytesMessage ) message;
            // Assume the length fits in an int - SQS only supports sizes up to 256k so
 that
            // should be true
            byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
            byteMessage.readBytes(bytes);
            System.out.println( "\t" +  Base64.encodeAsString( bytes ) );
        } else if( message instanceof ObjectMessage ) {
            ObjectMessage objMessage = (ObjectMessage) message;
            System.out.println( "\t" + objMessage.getObject() );
        }
    }
}
```

Amazon SQS でサポートされている JMS 1.1 の実装

Amazon SQS Javaメッセージングライブラリでは、次のJMS 1.1 実装がサポートされています。Amazon SQS Javaメッセージングライブラリでサポートされている機能と性能の詳細については、「Amazon SQS FAQ」を参照してください。

サポートされている共通インターフェース

- Connection
- ConnectionFactory
- Destination
- Session
- MessageConsumer
- MessageProducer

サポートされているメッセージタイプ

- ByteMessage

- ObjectMessage
- TextMessage

サポートされているメッセージ確認モード

- AUTO_ACKNOWLEDGE
- CLIENT_ACKNOWLEDGE
- DUPS_OK_ACKNOWLEDGE
- UNORDERED_ACKNOWLEDGE

> **Note**
>
> UNORDERED_ACKNOWLEDGEモードはJMS1.1仕様パートには含まれていません。このモードにより、JMSクライアントによるメッセージの明示的な確認を Amazon SQS が許可できるようになります。

JMS定義ヘッダーと予約プロパティ

メッセージの送信用

メッセージを送信する場合は、各メッセージに以下のヘッダーおよびプロパティを設定できます:

- JMSXGroupID (FIFOキューの場合は必須で、スタンドダードキューには許可されません)
- JMS_SQS_DeduplicationId(FIFO キューの場合はオプションで、 スタンダードキューには許可されません)

メッセージを送信すると、Amazon SQSにより各メッセージに以下のヘッダーおよびプロパティが設定されます:

- JMSMessageID
- JMS_SQS_SequenceNumber(FIFO キューの場合のみ)

メッセージの受信用

メッセージを受信すると、Amazon SQSにより各メッセージに以下のヘッダーおよびプロパティが設定されます:

- JMSDestination
- JMSMessageID
- JMSRedelivered
- JMSXDeliveryCount
- JMSXGroupID(FIFO キューの場合のみ)
- JMS_SQS_DeduplicationId(FIFO キューの場合のみ)
- JMS_SQS_SequenceNumber(FIFO キューの場合のみ)

Amazon SQSチュートリアル

このセクションでは、Amazon SQSの特徴と機能を検索するために使用できるチュートリアルを提供します。

トピック

- を使用した Amazon SQS キューの作成 AWS CloudFormation
- チュートリアル:Amazon Virtual Private Cloud から Amazon SQSキューにメッセージを送信する

を使用した Amazon SQS キューの作成 AWS CloudFormation

AWS CloudFormation コンソールと JSON (または YAML) テンプレートを使用して、Amazon SQS キューを作成できます。詳細については、AWS CloudFormation ユーザーガイドの「AWS CloudFormation テンプレート および AWS::SQS::Queue リソースでの作業」を参照してください。

AWS CloudFormation を使用して Amazon SQS キューを作成するには。

1. 次のJSON コードをMyQueue.jsonという名前のファイルにコピーします。スタンダードキューを作成するには、FifoQueue およびContentBasedDeduplicationプロパティを省略します。コンテンツベースの重複排除の詳細については、「Amazon SQS での 1 回限りの処理」を参照してください。

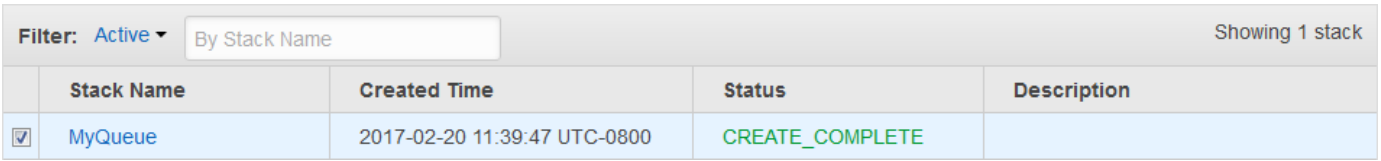
   > **Note**
   >
   > FIFOキューの名前は.fifoのサフィックスで終わる必要があります。

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "MyQueue": {
         "Properties": {
            "QueueName": "MyQueue.fifo",
            "FifoQueue": true,
            "ContentBasedDeduplication": true
             },
```

```
         "Type": "AWS::SQS::Queue"
         }
      },
   "Outputs": {
      "QueueName": {
         "Description": "The name of the queue",
         "Value": {
            "Fn::GetAtt": [
               "MyQueue",
               "QueueName"
            ]
         }
      },
      "QueueURL": {
         "Description": "The URL of the queue",
         "Value": {
            "Ref": "MyQueue"
         }
      },
      "QueueARN": {
         "Description": "The ARN of the queue",
         "Value": {
            "Fn::GetAtt": [
               "MyQueue",
               "Arn"
            ]
         }
      }
   }
}
```

2. AWS CloudFormation コンソール にサインインし、続いて [スタックの作成] を選択します。
3. [Specify Template] パネルで、[Upload a template file]、`MyQueue.json` ファイル、[次へ] の順に選択します。
4. {詳細を指定する} ページで、[`MyQueue`スタック名] に 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 は`MyQueue`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、 AWS CloudFormation に [CREATE_COMPLETE] ステータスが表示されます。



7. (オプション) キューの名前、URL、ARN を表示するには、スタックの名前を選択し、次のページで [出力] セクションを展開します。

チュートリアル:Amazon Virtual Private Cloud から Amazon SQS キューにメッセージを送信する

このチュートリアルでは、安全なプライベートネットワーク経由で Amazon SQSキューにメッセージを送信する方法について説明します。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Amazon SQSに接続し、インターフェイス VPC エンドポイントを使用して、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 インスタンスに接続して Amazon SQSキューにメッセージを送信できます。詳細については、「Amazon SQSのAmazon Virtual Private Cloud エンドポイント」を参照してください。

Important

- Amazon Virtual Private Cloud は HTTPS Amazon SQSエンドポイントでのみ使用できます。
- Amazon VPCからメッセージを送信するように Amazon SQS を設定する場合、プライベート DNS を有効にして、`sqs.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。

トピック

- ステップ1:Amazon EC2 キーペアを作成する
- ステップ 2: AWS リソースを作成する
- ステップ 3: EC2 インスタンスがパブリックアクセス可能ではないことを確認する
- ステップ4: Amazon SQSの Amazon VPC エンドポイントを作成する

- ステップ5:Amazon SQSキューにメッセージを送信する

ステップ1:Amazon EC2 キーペアを作成する

キーペアを使用すると、Amazon EC2 インスタンスに接続することができます。これは、ログイン情報を暗号化す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。

1. Amazon EC2 コンソールにサインインします。
2. ナビゲーションメニューの [ネットワーク & セキュリティ] で、[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. [キーペア作成] ダイアログボックスの [キーペア名] に `SQS-VPCE-Tutorial-Key-Pair` を入力し、[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`SQS-VPCE-Tutorial-Key-Pair.pem`が自動的にダウンロードされます。

   > **Important**
   >
   > このファイルを安全な場所に保存します。EC2は、2回目に同じキーペアに対して`.pem`ファイルを生成しません。

6. SSHクライアントに EC2 インスタンスへの接続を許可するには、相手のユーザーのみが読み取り権限を持つことができるように、プライベートキーファイルのアクセス許可を設定します。例:

   ```
   chmod 400 SQS-VPCE-Tutorial-Key-Pair.pem
   ```

ステップ 2: AWS リソースを作成する

必要なインフラストラクチャを設定するには、テンプレート を使用する必要があります。これは、Amazon EC2 AWS CloudFormation インスタンスや Amazon SQS キューなどの AWS リソースで構成されるスタックを作成するための設計図です。

このチュートリアルのスタックには、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サブネット内に起動されたAmazon EC2インスタンス
- Amazon SQSキュー

1. SQS-VPCE-Tutorial-CloudFormation.yaml から という名前の AWS CloudFormation テンプレートをダウンロードしますGitHub。
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、SQS-VPCE-SQS-Tutorial-CloudFormation.yamlファイルを選択して [次へ] を選択します。
5. [詳細の指定] ページで、以下の作業を行います。
   a. [スタック名] に「SQS-VPCE-Tutorial-Stack」と入力します。
   b. にはKeyName、SQS-VPCE-Tutorial-Key-Pair を選択します。
   c. [次へ] をクリックします。
6. [Options(オプション)] ページで、[Next(次へ)] を選択します。
7. 「確認」ページの「機能」セクションで、 がカスタム名で IAM リソースを作成する AWS CloudFormation 可能性があることを承認し、「 の作成」を選択します。

AWS CloudFormation は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、 AWS CloudFormation に [CREATE_COMPLETE]ステータスが表示されます。

ステップ 3: EC2 インスタンスがパブリックアクセス可能ではないことを確認する

AWS CloudFormation テンプレートは、VPC SQS-VPCE-Tutorial-EC2-Instanceに という名前の EC2 インスタンスを起動します。このEC2 インスタンスはアウトバウンドトラフィックを許可せず、 Amazon SQS にメッセージを送信することができません。これを確認するには、インスタンスに接続し、パブリックエンドポイントへの接続を試行してからメッセージAmazon SQSを送信してみる必要があります。

1. Amazon EC2 コンソールにサインインします。
2. ナビゲーションメニューで、[インスタンス] の下にある [インスタンス] を選択します。

3. SQS-VPCE-Tutorial-EC2Instance を選択します。
4. [パブリック DNS (IPv4)] の下でホスト名をコピーします。例、ec2-203-0-113-0.us-west-2.compute.amazonaws.com。
5. 先に作成したキーペアが格納されているディレクトリから次のコマンドを使用してインスタンスに接続します、例:

```
ssh -i SQS-VPCE-Tutorial-Key-Pair.pem ec2-user@ec2-203-0-113-0.us-east-2.compute.amazonaws.com
```

6. パブリックエンドポイントに接続を試みます、例:

```
ping amazon.com
```

接続の試行は予期したとおりに失敗します。

7. Amazon SQSコンソールにサインインします。
8. キューのリストから、VPCE-SQS-Tutorial-Stack-CFQueue-1ABCDEFGH2IJK など、 AWS CloudFormation テンプレートによって作成されたキューを選択します。
9. 詳細 テーブルで、URL をコピーします、例、 https://sqs.us-east-2.amazonaws.com/123456789012/。
10. EC2 インスタンスから、次のコマンドを使用して、キューにメッセージを発行してみます、例:

```
aws sqs send-message --region us-east-2 --endpoint-url https://sqs.us-east-2.amazonaws.com/ --queue-url https://sqs.us-east-2.amazonaws.com/123456789012/
 --message-body "Hello from Amazon SQS."
```

送信の試行は予期したとおりに失敗します。

> **Important**
>
> 後で Amazon SQSのVPC エンドポイントを作成するときに、送信の試行は成功します。

ステップ4: Amazon SQSの Amazon VPC エンドポイントを作成する

VPCを Amazon SQS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後、VPC内のEC2 インスタンスから Amazon SQS API を使用できます。これに

より、パブリックインターネットを経由せずに、 AWS ネットワーク内のキューにメッセージを送信できます。

> **Note**
>
> EC2 インスタンスは、インターネット上の他の AWS サービスとエンドポイントには引き続きアクセスできません。

1. Amazon VPC コンソールにサインインします。
2. 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 [サービス名] で、 Amazon SQS のサービス名を選択します。

   > **Note**
   >
   > サービス名は、現在の AWS リージョンによって異なります。たとえば、米国東部 (オハイオ) にいる場合、サービス名は ***com.amazonaws.***us-east-2.sqs になります。

5. [VPC] には、SQS-VPCE-Tutorial-VPC を選択します。
6. [サブネット] には、[サブネット ID] にSQS-VPCE-Tutorial-Subnetを含むサブネットを選択します。
7. [セキュリティグループ] の場合は [セキュリティグループの選択] を選択し、[グループ名] にSQS VPCE Tutorial Security Groupを含むセキュリティグループを選択します。
8. [エンドポイントの作成] を選択します。

   インターフェイス VPC エンドポイントが作成され、その ID が表示されます。例: vpce-0ab1cdef2ghi3j456k。
9. [閉じる] 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。

Amazon VPCがエンドポイントの作成を開始し、{保留中] ステータスが表示されます。プロセスが完了すると、Amazon VPCに {利用可能} ステータスが表示されます。

ステップ5:Amazon SQSキューにメッセージを送信する

これで VPCに Amazon SQSのエンドポイントが含まれたので、EC2 インスタンスに接続して、キューにメッセージを送信できます。

1. EC2インスタンスに再接続します、例:

```
ssh -i SQS-VPCE-Tutorial-Key-Pair.pem ec2-user@ec2-203-0-113-0.us-
east-2.compute.amazonaws.com
```

2. 次のコマンドを使用して、もう一度キューにメッセージを発行してみます、例:

```
aws sqs send-message --region us-east-2 --endpoint-url https://sqs.us-
east-2.amazonaws.com/ --queue-url https://sqs.us-east-2.amazonaws.com/123456789012/
 --message-body "Hello from Amazon SQS."
```

送信の試行が成功し、メッセージ本文のMD5ダイジェストとメッセージ ID が表示されます、例:

```
{
 "MD5OfMessageBody": "a1bcd2ef3g45hi678j90klmn12p34qr5",
 "MessageId": "12345a67-8901-2345-bc67-d890123e45fg"
}
```

AWS CloudFormation テンプレートによって作成されたキュー (VPCE-SQS-Tutorial-Stack-CFQueue-1ABCDEFGH2IJK など) からのメッセージの受信と削除については、「」を参照してくださいAmazon SQS でのメッセージの受信と削除 。

リソースの削除の詳細については、以下を参照してください。

- VPC エンドポイントの削除のAmazon VPC User Guide
- Amazon SQSキューの削除
- 「Amazon EC2 ユーザーガイド」の「インスタンスの終了Amazon EC2」
- VPC を削除するのAmazon VPC User Guide
- AWS CloudFormation ユーザーガイドの AWS CloudFormation コンソールでのスタックの削除
- 「Amazon EC2 ユーザーガイド」の「キーペアの削除Amazon EC2」

Amazon SQS の問題のトラブルシューティング

以下のトピックでは、Amazon SQS コンソール、Amazon SQS API、または Amazon SQS で他のツールを使用する際に発生する可能性がある一般的なエラーや問題のトラブルシューティングに関するアドバイスを提供します Amazon SQS 。ここに記載されていない問題が見つかった場合は、このページの [Feedback] ボタンを使用して報告することができます。

トラブルシューティングに関するアドバイス、およびサポートへの一般的な質問に対する回答については、AWS ナレッジセンターにアクセスしてください。

トピック

- Amazon SQS でアクセス拒否エラーのトラブルシューティング
- Amazon SQS API エラーのトラブルシューティング
- Amazon SQS デッドレターキューと DLQ リドライブの問題のトラブルシューティング
- Amazon SQS での FIFO スロットリングの問題のトラブルシューティング
- Amazon SQS API ReceiveMessage コールで返されないメッセージのトラブルシューティング
- Amazon SQS ネットワークエラーのトラブルシューティング
- Amazon Simple Queue Serviceのキュー AWS X-Rayを使用したトラブルシューティング

Amazon SQS でアクセス拒否エラーのトラブルシューティング

以下のトピックでは、Amazon SQS API コールの `AccessDenied`または `AccessDeniedException` エラーの最も一般的な原因について説明します。これらのエラーのトラブルシューティング方法の詳細については、 AWS ナレッジセンターガイドのAmazon SQS API コールのAccessDenied「」またはAccessDenied「例外」エラーのトラブルシューティング方法」を参照してください。

エラーメッセージの例：

```
An error occurred (AccessDenied) when calling the SendMessage operation: Access to
        the resource https://sqs.us-east-1.amazonaws.com/ is denied.
```

- または -

```
An error occurred (KMS.AccessDeniedException) when calling the SendMessage
```

```
        operation: User: arn:aws:iam::xxxxx:user/xxxx is not authorized to perform:
        kms:GenerateDataKey on resource: arn:aws:kms:us-east-1:xxxx:key/xxxx with an
explicit
        deny.
```

トピック

- Amazon SQS キューポリシーと IAM ポリシー
- AWS Key Management Service アクセス許可
- VPC エンドポイントポリシー
- 組織のサービスコントロールポリシー

Amazon SQS キューポリシーと IAM ポリシー

リクエスタが Amazon SQS オペレーションを実行するための適切なアクセス許可を持っているかどうかを確認するには、次の手順を実行します。

- Amazon SQS API コールを実行している IAM プリンシパルを特定します。IAM プリンシパルが同じアカウントからのものである場合、Amazon SQS キューポリシーまたは AWS Identity and Access Management (IAM) ポリシーのいずれかに、 アクションへのアクセスを明示的に許可するアクセス許可を含める必要があります。
- プリンシパルが IAM エンティティの場合：
  - IAM ユーザーまたはロールを識別するには、 の右上隅を確認するか AWS Management Console、 `aws sts get-caller-identity` コマンドを使用します。
  - IAM ユーザーまたはロールに関連付けられている IAM ポリシーを確認します。次のいずれかの方法を使用します。
    - IAM Policy Simulator を使用して IAM ポリシーをテストします。
    - さまざまな IAM ポリシータイプを確認します。
  - 必要に応じて、IAM ユーザーポリシーを編集します。
  - キューポリシーを確認し、必要に応じて編集します。
- プリンシパルが AWS サービスの場合、Amazon SQS キューポリシーはアクセスを明示的に許可する必要があります。
- プリンシパルがクロスアカウントプリンシパルの場合、Amazon SQS キューポリシーと IAM ポリシーの両方がアクセスを明示的に許可する必要があります。
- ポリシーが条件要素を使用している場合は、条件がアクセスを制限することを確認します。

> **Important**
>
> どちらのポリシーでも明示的に拒否すると、明示的な許可が上書きされます。Amazon SQS ポリシー の基本的な例を次に示します。

AWS Key Management Service アクセス許可

Amazon SQS キューで、カスタマー管理の でサーバー側の暗号化 (SSE) が有効になっている場合は AWS KMS key、プロデューサーとコンシューマーの両方にアクセス許可を付与する必要があります。キューが暗号化されているかどうかを確認するには、 `GetQueueAttributes` API `KmsMasterKeyId` 属性を使用するか、暗号化 のキューコンソールから使用できます。

- プロデューサーに必要なアクセス許可 :

```
{
"Effect": "Allow",
"Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
],
"Resource": "<Key ARN>"
}
```

- コンシューマーに必要なアクセス許可 :

```
{
"Effect": "Allow",
"Action": [
    "kms:Decrypt"
],
"Resource": "<Key ARN>"
}
```

- クロスアカウントアクセスに必要なアクセス許可 :

```
{
"Effect": "Allow",
"Action": [
    "kms:DescribeKey",
    "kms:Decrypt",
```

```
    "kms:ReEncrypt",
    "kms:GenerateDataKey"
],
"Resource": "<Key ARN>"
}
```

Amazon SQS キューの暗号化を有効にするには、次のいずれかを使用できます。

- SSE-Amazon SQS (Amazon SQS サービスによって作成および管理される暗号化キー)
- AWS マネージドデフォルトキー (alias/aws/sqs)
- カスタマーマネージドキー

ただし、 AWS管理の KMS キー を使用している場合、デフォルトのキーポリシーを変更することはできません。したがって、他の サービスやクロスアカウントへのアクセスを提供するには、カスタマーマネージドキーを使用します。これにより、キーポリシーを編集できます。

VPC エンドポイントポリシー

Amazon SQS Amazon Virtual Private Cloud (Amazon VPC) エンドポイント を介して Amazon SQS にアクセスする場合、Amazon SQS VPC エンドポイントポリシーはアクセスを許可する必要があります。Amazon SQS の Amazon VPC エンドポイントのポリシーを作成できます。このポリシーでは、以下を指定できます。

1. アクションを実行できるプリンシパル。
2. 実行可能なアクション。
3. このアクションを実行できるリソース。

次の例では、VPC エンドポイントポリシー*MyUser*は、IAM ユーザーが Amazon SQS キュー にメッセージを送信することを許可することを指定します*MyQueue*。その他のアクション、IAM ユーザー、および Amazon SQS リソースは、VPC エンドポイントを介したアクセスを拒否されます。

```
{
   "Statement": [{
      "Action": ["sqs:SendMessage"],
      "Effect": "Allow",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Principal": {
```

```
        "AWS": "arn:aws:iam:123456789012:user/MyUser"
      }
   }]
}
```

組織のサービスコントロールポリシー

が組織に AWS アカウント 属している場合、 AWS Organizations ポリシーによって Amazon SQS キューへのアクセスがブロックされる可能性があります。デフォルトでは、 AWS Organizations ポリシーは Amazon SQS へのリクエストをブロックしません。ただし、 AWS Organizations ポリシーが Amazon SQS キューへのアクセスをブロックするように設定されていないことを確認してください。 AWS Organizations ポリシーを確認する手順については、「 AWS Organizations ユーザーガイド」の「すべてのポリシーの一覧表示」を参照してください。

Amazon SQS API エラーのトラブルシューティング

以下のトピックでは、Amazon SQS API コールを行うときに返される最も一般的なエラーとそのトラブルシューティング方法について説明します。

トピック

- QueueDoesNotExist エラー
- InvalidAttributeValue エラー
- ReceiptHandle エラー

QueueDoesNotExist エラー

このエラーは、Amazon SQS サービスが Amazon SQS アクションの前述のキューを見つけられない場合に返されます。

考えられる原因と緩和策：

- リージョンが正しくない： Amazon SQS クライアント設定を確認して、クライアントで正しいリージョンが設定されていることを確認します。クライアントでリージョンを設定しない場合、SDK または は設定ファイルまたは環境変数からリージョン AWS CLI を選択します。SDK が設定ファイルにリージョンを見つけられない場合、SDK はデフォルトでリージョンを us-east-1 に設定します。

- キューが最近削除される可能性がある： API コールが行われる前にキューが削除された場合、API コールはこのエラーを返します。エラーが発生する前にDeleteQueue、オペレーションがない CloudTrailか確認します。
- アクセス許可の問題： リクエスト AWS Identity and Access Management 元の (IAM) ユーザーまたはロールに必要なアクセス許可がない場合、次のエラーが表示されることがあります。

  ```
  The specified queue does not exist or you do not have access to it.
  ```

  アクセス許可を確認し、正しいアクセス許可で API コールを行います。

`QueueDoesNotExist` エラーのトラブルシューティングの詳細については、「 AWS ナレッジセンターガイド」のAmazon SQSキューに API コールを行うときに QueueDoesNotExist エラーをトラブルシューティングする方法」を参照してください。

InvalidAttributeValue エラー

このエラーは、Amazon SQS キューリソースポリシー、または誤ったポリシーまたはプリンシパルを持つプロパティを更新すると返されます。

考えられる原因と緩和策：

- 無効なリソースポリシー： リソースポリシーにすべての必須フィールドがあることを確認します。詳細については、「IAM JSON ポリシー要素のリファレンス」および「IAM ポリシーの検証」を参照してください。IAM Policy Generator を使用して、Amazon SQS リソースポリシーを作成およびテストすることもできます。ポリシーが JSON 形式であることを確認します。
- 無効なプリンシパル： `Principal`要素がリソースポリシーに存在し、値が有効であることを確認します。Amazon SQS リソースポリシー`Principal`要素に IAM エンティティが含まれている場合は、ポリシーを使用する前にエンティティが存在することを確認してください。Amazon SQS はリソースポリシーを検証し、IAM エンティティをチェックします。IAM エンティティが存在しない場合は、エラーが表示されます。IAM エンティティを確認するには、 `GetRole`および `GetUser` APIsを使用します。

`InvalidAttributeValue` エラーのトラブルシューティング方法の詳細については、「 AWS ナレッジセンターガイド」のAmazon SQSコールを行うときに QueueDoesNotExist エラーをトラブルシューティングする方法」を参照してください。

ReceiptHandle エラー

DeleteMessage API コールを行うと、受信ハンドルが正しくないか、期限切れになると、エラーまたは が返されReceiptHandleIsInvalidInvalidParameterValueる可能性があります。

- ReceiptHandleIsInvalid error: 受信ハンドルが正しくない場合は、次の例のようなエラーが表示されます。

```
An error occurred (ReceiptHandleIsInvalid) when calling the DeleteMessage operation:
 The input receipt handle <YOUR RECEIPT HANDLE> is not a valid receipt handle.
```

- InvalidParameterValue error: 受信ハンドルの有効期限が切れている場合、次の例のようなエラーが表示されます。

```
An error occurred (InvalidParameterValue) when calling the DeleteMessage operation:
 Value <YOUR RECEIPT HANDLE> for parameter ReceiptHandle is invalid. Reason: The
 receipt handle has expired.
```

考えられる原因と緩和策：

受信ハンドルは受信したメッセージごとに作成され、可視性タイムアウト期間にのみ有効です。可視性タイムアウト期間が終了すると、コンシューマーのキューにメッセージが表示されます。コンシューマーからメッセージを再度受信すると、新しい受信ハンドルを受け取ります。受信ハンドルのエラーが正しくないか、期限切れにならないようにするには、正しい受信ハンドルを使用して、Amazon SQS キューの可視性タイムアウト期間内にメッセージを削除します。

ReceiptHandle エラーのトラブルシューティング方法の詳細については、「 AWS ナレッジセンターガイド」のAmazon SQS DeleteMessage コールを使用する際の」エラーReceiptHandleIsInvalidとInvalidParameter「値」エラーのトラブルシューティング方法」を参照してください。

Amazon SQS デッドレターキューと DLQ リドライブの問題のトラブルシューティング

以下のトピックでは、Amazon SQS DLQ および DLQ リドライブの問題の最も一般的な原因と、それらのトラブルシューティング方法について説明します。詳細については、「 ナレッジセンターガイドAmazon SQS DLQ リドライブの問題のトラブルシューティング方法」を参照してください。AWS

トピック

DLQ の問題

一般的な DLQ の問題とその解決方法について説明します。

トピック

コンソールを使用してメッセージを表示すると、メッセージがデッドレターキューに移動されることがあります。

Amazon SQSは、対応するキューの リドライブ ポリシーに対して コンソールでのメッセージの表示回数をカウントします。したがって、対応するキューのリドライブポリシーで指定された回数だけコンソールでメッセージを表示すると、メッセージは対応するキューのデッドレターキューに移動されます。

この動作を調整するには、次のオプションがあります。

- 対応するキューの Redrive ポリシーで、[Maximum Receives] 設定の値を大きくします。
- 対応するキューのメッセージがコンソールに表示されないようにします。

デッドレターキューの **NumberOfMessagesSent** と **NumberOfMessagesReceived** が一致しない

手動でデッドレターキューに送信したメッセージは、`NumberOfMessagesSent` メトリクスによってキャプチャされます。ただし、処理の試行が失敗した結果としてデッドレターキューにメッセージが送信された場合、メトリクスによってキャプチャされません。したがって、`NumberOfMessagesSent`と の値`NumberOfMessagesReceived`が異なる可能性があります。

デッドレターキューリドライブの作成と設定

デッドレターキューのリドライブでは、Amazon SQS がデッドレターキューからメッセージを受信し、宛先キューにメッセージを送信するための適切なアクセス許可を設定する必要があります。正しいアクセス許可がない場合、デッドレターキューのリドライブタスクが失敗する可能性があります。メッセージリドライブタスクのステータスを表示して問題を修正し、もう一度試してください。

標準および FIFO キューメッセージの失敗処理

標準キューは、保持期間 が終了するまでメッセージを処理し続けます。この継続的な処理により、キューが消費されていないメッセージによってブロックされる可能性が最小限に抑えられます。コンシューマーが繰り返し削除に失敗するメッセージが多数あると、コストが増加し、ハードウェアに余分な負荷がかかる可能性があります。コストを抑えるには、失敗したメッセージをデッドレターキューに移動します。

スタンダードキューでは、多数の未処理のメッセージも許可されます。メッセージの大部分が消費できず、デッドレターキューに送信されない場合、メッセージの処理速度が低下する可能性があります。キューの効率を維持するには、アプリケーションがメッセージ処理を正しく処理していることを確認してください。

FIFOキューは、メッセージグループからメッセージを順番に送信することで、ジャストワンス処理を実現します。したがって、コンシューマーは順序付けられたメッセージを別のメッセージグループから引き続き取得できますが、キューをブロックするメッセージが正常に処理されるか、デッドレターキューに移動するまで、最初のメッセージグループは使用できなくなります。

さらに、FIFO キューでは、処理中のメッセージの数を減らすことができます。FIFO キューがメッセージによってブロックされないようにするには、アプリケーションがメッセージ処理を正しく処理していることを確認してください。

詳細については、「Amazon SQS メッセージクォータ」および「Amazon SQSメッセージの操作」を参照してください。

DLQ リドライブの問題

一般的な DLQ リドライブの問題とその解決方法について説明します。

トピック

- AccessDenied アクセス許可の問題
- NonExistentQueue エラー

- CouldNotDetermineMessageソースエラー

AccessDenied アクセス許可の問題

このAccessDeniedエラーは、 AWS Identity and Access Management (IAM) エンティティに必要なアクセス許可がないために DLQ リドライブが失敗した場合に発生します。

エラーメッセージの例：

```
Failed to create redrive task. Error code: AccessDenied - Queue Permissions to Redrive.
```

DLQ リドライブリクエストを行うには、次の API アクセス許可が必要です。

メッセージのリドライブを開始するには：

- デッドレターキューのアクセス許可：
  - sqs:StartMessageMoveTask
  - sqs:ReceiveMessage
  - sqs:DeleteMessage
  - sqs:GetQueueAttributes
  - kms:Decrypt – デッドレターキューまたは元のソースキューが暗号化されている場合。
- 送信先キューのアクセス許可：
  - sqs:SendMessage
  - kms:GenerateDataKey – 送信先キューが暗号化されている場合。
  - kms:Decrypt – 送信先キューが暗号化されている場合。

進行中のメッセージリドライブをキャンセルするには：

- デッドレターキューのアクセス許可：
  - sqs:CancelMessageMoveTask
  - sqs:ReceiveMessage
  - sqs:DeleteMessage
  - sqs:GetQueueAttributes
  - kms:Decrypt – デッドレターキューまたは元のソースキューが暗号化されている場合。

メッセージの移動ステータスを表示するには：

- デッドレターキューのアクセス許可：
  - `sqs:ListMessageMoveTasks`
  - `sqs:GetQueueAttributes`

NonExistentQueue エラー

このNonExistentQueueエラーは、Amazon SQS ソースキューが存在しないか、削除された場合に発生します。存在する Amazon SQS キューを確認してリドライブします。

エラーメッセージの例：

```
Failed: AWS.SimpleQueueService.NonExistentQueue
```

CouldNotDetermineMessageソースエラー

このCouldNotDetermineMessageSourceエラーは、次のシナリオで DLQ リドライブを開始しようとしたときに発生します。

- SendMessage API を使用して DLQ に直接送信される Amazon SQS メッセージ。
- DLQ が設定された Amazon Simple Notification Service (Amazon SNS) トピックまたは AWS Lambda 関数からのメッセージ。

このエラーを解決するには、リドライブを開始するときにカスタム送信先へのリドライブを選択します。次に、Amazon SQS キュー ARN を入力して、DLQ から送信先キューにすべてのメッセージを移動します。

エラーメッセージの例：

```
Failed: CouldNotDetermineMessageSource
```

Amazon SQS での FIFO スロットリングの問題のトラブルシューティング

デフォルトでは、FIFO キューは、、、および の API アクションごとに 1 秒あたり 300 SendMessage件のトランザクションをサポートしますReceiveMessageDeleteMessage。キュー

内のメッセージが利用可能な場合でも、300 TPS を超えるリクエストはThrottlingException工ラーを受け取ります。これを軽減するには、次の方法を使用できます。

- Amazon SQS で FIFO キューの高スループットを有効にする.
- Amazon SQS API バッチアクション SendMessageBatch、DeleteMessageBatch、および ChangeMessageVisibilityBatchを使用して、API アクションごとに 1 秒あたり最大 3,000 メッセージという TPS 制限を引き上げ、コストを削減します。ReceiveMessage API の場合、トランザクションごとに最大 10 個のメッセージを受信するように MaxNumberofMessagesパラメータを設定します。詳細については、「Amazon SQSのバッチアクション」を参照してください。
- スループットの高い FIFO キューの場合は、推奨事項に従ってパーティション使用率を最適化します。同じメッセージグループ IDsで送信します。メッセージを削除するか、同じ ReceiveMessage API リクエストからの受信ハンドルを使用してメッセージの可視性タイムアウト値をバッチで変更します。
- 一意のMessageGroupId値の数を増やします。これにより、FIFO キューパーティション間で均等に分散できます。詳細については、「Amazon SQSメッセージグループ ID の使用」を参照してください。

詳細については、「 ナレッジセンターガイド」のAmazon SQS FIFO キューが他のメッセージグループ内のすべてのメッセージまたはメッセージを返さないのはなぜですか？」を参照してください。 AWS

Amazon SQS API ReceiveMessage コールで返されないメッセージのトラブルシューティング

以下のトピックでは、Amazon SQS メッセージがコンシューマーに返されない最も一般的な原因と、そのトラブルシューティング方法について説明します。詳細については、「 AWS ナレッジセンターガイド」のAmazon SQSキューからメッセージを受信できないのはなぜですか？」を参照してください。

トピック

- キューを空にする
- 飛行中の制限に達しました
- メッセージ遅延
- メッセージは処理中です

- ポーリング方法

キューを空にする

キューが空かどうかを判断するには、ロングポーリングを使用して ReceiveMessage API を呼び出しま
す。、ApproximateNumberOfMessagesVisible、ApproximateNumberOfMessagesNotVisibleおよび ApproximateNumberOfMessagesDelayed CloudWatch メトリクスを使用することもできます。すべてのメトリクス値が数分間 0 に設定されている場合、キューは空と見なされます。

飛行中の制限に達しました

ロングポーリングを使用し、キューの処理中の制限 (FIFO の場合は 20000、デフォルトでは標準の場合は 120000) に違反した場合、Amazon SQS はクォータ制限 を超えるエラーメッセージを返しません。

メッセージ遅延

Amazon SQS キューが遅延キュー として設定されている場合、またはメッセージがメッセージタイマー で送信された場合、遅延時間が終わるまでメッセージは表示されません。キューが遅延キューとして設定されているかどうかを確認するには、 GetQueueAttributes API DelaySeconds 属性を使用するか、配信遅延 のキューコンソールから を使用します。ApproximateNumberOfMessagesDelayed CloudWatch メトリクスをチェックして、メッセージが遅延していないかどうかを確認します。

メッセージは処理中です

別のコンシューマーがメッセージをポーリングした場合、メッセージは可視性タイムアウト期間中は処理中または表示されません。追加のポーリングは、空の受信を返す場合があります。ApproximateNumberOfMessages表示メトリクス CloudWatchをチェックして、受信可能なメッセージの数を確認します。FIFO キューの場合、メッセージグループ ID を持つメッセージが処理中の場合、メッセージを削除しない限り、メッセージは返されません。そうしないと表示されなくなります。これは、メッセージの順序が FIFO キューのメッセージグループレベルで維持されるためです。

ポーリング方法

ショートポーリング を使用している場合、 (WaitTime秒は 0) Amazon SQS はサーバーのサブセットをサンプリングし、それらのサーバーからのメッセージのみを返します。したがって、 が受信でき

る場合でも、メッセージは取得されない場合があります。それ以降のポーリングリクエストはメッセージを返します。

ロングポーリング を使用している場合、Amazon SQS はすべてのサーバーをポーリングし、少なくとも 1 つの使用可能なメッセージを収集した後、指定された最大数までレスポンスを送信します。ReceiveMessage WaitTimeSeconds の値が低すぎると、使用可能なメッセージの一部が受信されない場合があります。

Amazon SQS ネットワークエラーのトラブルシューティング

以下のトピックでは、Amazon SQS のネットワーク問題の最も一般的な原因と、それらのトラブルシューティング方法について説明します。

トピック

ETIMEOUT error

ETIMEOUT エラーは、クライアントが Amazon SQS エンドポイントへの TCP 接続を確立できない場合に発生します。

トラブルシューティング：

- ネットワーク接続を確認する

  などのコマンドを実行して、Amazon SQS へのネットワーク接続をテストしますtelnet。

  Example: telnet sqs.us-east-1.amazonaws.com 443
- ネットワーク設定の確認
  - ローカルファイアウォールのルール、ルート、アクセスコントロールリスト (ACLsで、使用するポートでのトラフィックが許可されていることを確認します。
  - セキュリティグループのアウトバウンド (エグレス) ルールでは、ポート 80 または 443 へのトラフィックを許可する必要があります。
  - ネットワーク ACL アウトバウンド (送信) ルールでは、TCP ポート 80 または 443 へのトラフィックを許可する必要があります。

  - ネットワーク ACL のインバウンド (受信) ルールでは、TCP ポート 1024-65535 でのトラフィックを許可する必要があります。
  - パブリックインターネットに接続する Amazon Elastic Compute Cloud (Amazon EC2) インスタンスには、インターネット接続 が必要です。
- Amazon Virtual Private Cloud (Amazon VPC) エンドポイント

  Amazon VPC エンドポイントを介して Amazon SQS にアクセスする場合、エンドポイントセキュリティグループは、ポート 443 でクライアントセキュリティグループへのインバウンドトラフィックを許可する必要があります。VPC エンドポイントのサブネットに関連付けられたネットワーク ACL には、次の設定が必要です。
  - ネットワーク ACL アウトバウンド (エグレス) ルールでは、TCP ポート 1024-65535 (エフェメラルポート) でのトラフィックを許可する必要があります。
  - ネットワーク ACL のインバウンド (進入) ルールでは、ポート 443 でのトラフィックを許可する必要があります。

また、Amazon SQS VPC エンドポイント AWS Identity and Access Management (IAM) ポリシーではアクセスを許可する必要があります。次の VPC エンドポイントポリシーの例では、IAM ユーザーが Amazon SQS キュー にメッセージを送信*MyUser*できるように指定しています*MyQueue*。その他のアクション、IAM ユーザー、および Amazon SQS リソースは、VPC エンドポイントを介したアクセスを拒否されます。

```
{
    "Statement": [{
        "Action": ["sqs:SendMessage"],
        "Effect": "Allow",
        "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
        "Principal": {
            "AWS": "arn:aws:iam:123456789012:user/MyUser"
        }
    }]
}
```

UnknownHostException error

UnknownHostException エラーは、ホスト IP アドレスを特定できなかった場合に発生します。

トラブルシューティング：

nslookup ユーティリティを使用して、ホスト名に関連付けられた IP アドレスを返します。

- Windows and Linux OS

```
nslookup sqs.<region>.amazonaws.com
```

- AWS CLI または SDK for Python のレガシーエンドポイント：

```
nslookup <region>.queue.amazonaws.com
```

失敗した出力を受け取った場合は、「 AWS ナレッジセンターガイド」の「DNS の仕組み」および「DNS の部分的または断続的な障害のトラブルシューティング方法」の指示に従ってください。

有効な出力を受け取った場合は、アプリケーションレベルの問題である可能性があります。アプリケーションレベルの問題を解決するには、次の方法を試してください。

- アプリケーションを再起動します。
- Java アプリケーションに不正な DNS キャッシュがないことを確認します。可能であれば、DNS TTL に準拠するようにアプリケーションを設定します。詳細については、「DNS 名ルックアップの JVM TTL の設定」を参照してください。

ネットワークエラーのトラブルシューティング方法の詳細については、 AWS ナレッジセンターガイドのAmazon SQS の「ETIMEOUT」およびUnknownHost「Exception」接続エラーのトラブルシューティング方法」を参照してください。

Amazon Simple Queue Serviceのキュー AWS X-Rayを使用した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、データを表示およびフィルタリングして潜在的な問題や最適化の機会を特定できます。アプリケーションへのトレースされたリクエストについては、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 APIs に対して行う呼び出しに関する詳細情報を確認できます。

Amazon SQS を介して AWS X-Ray トレースヘッダーを送信するには、次のいずれかを実行します。

- `X-Amzn-Trace-Id` トレースヘッダーを使用する。
- `AWSTraceHeader` メッセージシステム属性を使用する。

エラーとレイテンシーに関するデータを収集するには、`AmazonSQS`AWS X-Ray SDKを使用してクライアントをインスツルメンテーションする必要があります。

AWS X-Ray コンソールを使用して、Amazon SQS とアプリケーションが使用する他の 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「 を Amazon SQS AWS X-Rayに使用する」を参照してください。

Amazon SQSのセキュリティ

このセクションでは、Amazon SQSのセキュリティ、認証、アクセスコントロールおよびAmazon SQSアクセスポリシー言語について説明します。

トピック

Amazon SQS でのデータ保護

責任 AWS 共有モデル、Amazon Simple Queue 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 はすべての 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があります AWS クラウド。お客様は、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管理を維持する責任があります。また、使用する AWS のサービス のセキュリティ設定と管理タスクもユーザーの責任となります。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。

データ保護の目的で、 認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定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権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 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 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。

- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密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イスまたは API AWS を介して 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。利用可能な 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。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキスト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、または SDK を使用して Amazon SQS AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs 名前に使用する自由記述のテキストフィールドやタグ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。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

次のセクションでは、Amazon SQSのデータ保護について説明します。

トピック

- Amazon SQS でのデータ暗号化
- Amazon SQS でのネットワークトラフィックのプライバシー

Amazon SQS でのデータ暗号化

データ保護には、転送時 (Amazon SQS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 SQSデータセンターのディスクに保管されているとき) 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 SQS 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。保管中のデータは、Amazon SQS にメッセージを保存してから、その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ようリクエストすることで保護できます。Amazon SQS では、データ暗号化を最適化するために SSE の使用を推奨しています。

トピック

- Amazon SQS での保管時の暗号化
- Amazon SQS キー管理

Amazon SQS での保管時の暗号化

サーバー側の暗号化 (SSE)では、機密データを暗号化されたキューで送信できます。SSE は、SQS 管理の暗号化キー (SSE-SQS) または AWS Key Management Service (SSE-KMS) で管理されるキーを使用して、キュー内のメッセージの内容を保護します。を使用して SSE を管理する方法については AWS Management Console、以下を参照してください。

- キューのSSE-SQSの設定(コンソール)
- キューのSSE-KMSの設定(コンソール)

(および 、`CreateQueue`、 `GetQueueAttributes`アクション) を使用した AWS SDK for Java SSE の管理については`SetQueueAttributes`、次の例を参照してください。

- Amazon SQS キューでのサーバー側の暗号化の使用
- の KMS アクセス許可の設定 AWS のサービス

Amazon SQS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、承認済みのコンシューマーに送信された場合のみ、Amazon SQSによって解読されます。

> **Important**
>
> SSE が有効なキューへのすべて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。
> デフォルトキー (Amazon SQS のAWS マネージド KMS キー) を使用する暗号化されたキューは、別の で Lambda 関数を呼び出すことはできません AWS アカウント。 Amazon SQS
> AWS Security Token Service `AssumeRole` アクションを使用して Amazon SQSに通知を送信できる AWS サービスの一部の機能は、SSE と互換性がありますが、標準キューでのみ動作します。
>
> - Auto Scalingライフサイクルフック
> - AWS Lambda デッドレターキュー

暗号化されたキューとの他のサービスとの互換性については、AWS サービスの KMS アクセス許可を設定するとサービスドキュメント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。で Amazon SQS を使用すると AWS KMS、メッセージデータを暗号化するデータキーも暗号化され、保護するデータとともに保存されます。

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 keys を作成および管理できます。
- Amazon SQS の AWS マネージド KMS キーを使用することもできます。これは、アカウントとリージョンごとに一意です。
- AWS KMS 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のに役立ちます。

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 SQS キューのメッセージ本文が暗号化されます。

SSEでは、以下は暗号化されません。

- キューのメタデータ (キュー名と属性)
- メッセージのメタデータ (メッセージ ID、タイムスタンプ、属性)
- キューごとのメトリクス

メッセージを暗号化すると、未承認ユーザーまたは匿名ユーザーはそのメッセージを利用できなくなります。SSE を有効にすると、暗号化されたキューへの匿名の `SendMessage` リクエストと `ReceiveMessage` リクエストは拒否されます。Amazon SQS セキュリティベストプラクティスで

は、匿名リクエストを使用しないことを推奨しています。Amazon SQS キューに匿名リクエストを送信する場合は、必ず SSE を無効にしてください。これは、Amazon SQSの正常な機能には影響しません。

- メッセージが暗号化されるのは、キュー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 SQSは、バックログされたメッセージを暗号化しません。
- キュー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。

メッセージをデッドレターキューに移動しても、暗号化には影響しません。

- 暗号化された出典キューから暗号化されていないデッドレターキューにメッセージがAmazon SQS によって移動された場合も、メッセージは暗号化された状態で維持されます。
- 暗号化されていない出典キューから暗号化されたデッドレターキューにメッセージがAmazon SQSによって移動された場合、メッセージは暗号化されていない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 キューサ - ビス API リファレンス」を参照してください。

データキー

キー (DEK) は、Amazon SQSメッセージの内容を暗号化します。

詳細については、AWS Encryption SDK デベロッパーガイド。の「AWS Key Management Service デベロッパーガイドのデータキー」を参照してください。

データキー再利用期間

Amazon SQS がデータキーを再利用してメッセージを暗号化または復号してから AWS KMS 、再度 を呼び出すことができる秒単位の時間の長さ。60秒(1分)から86,400秒(24時間)の秒数を表す整数。デフォルトは300(5分)です。詳細については、「データキー再利用期間について」を参照してください。

> **Note**
>
> 万一 に到達できない場合 AWS KMS、Amazon SQS は接続が再確立されるまでキャッシュされたデータキーを使用し続けます。

KMS キー ID

アカウント内または別のアカウントの AWS マネージド KMS キーまたはカスタム KMS キーのエイリアス、エイリアス ARN、キー ID、またはキー ARN。Amazon SQS の AWS マネージド KMS キーのエイリアスは常に ですが`alias/aws/sqs`、カスタム KMS キーのエイリアスは などです`alias/MyAlias`。これらの KMS キーで Amazon SQS キュー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。
>
> - カスタム KMS キーを指定しない場合、Amazon SQS は Amazon SQS の AWS マネージド KMS キーを使用します。 Amazon SQS
> - AWS Management Console を使用してキューの Amazon SQS の AWS マネージド KMS キーを初めて指定すると、 AWS KMS は Amazon SQS の AWS マネージド KMS キーを作成します。 Amazon SQS
> - または、SSE が有効になっているキューで `SendMessage`または `SendMessageBatch`アクションを初めて使用すると、 は Amazon SQS の AWS マネージド KMS キー AWS KMS を作成します。 Amazon SQS

KMS キーの作成、KMS キーの使用方法を制御するポリシーの定義、およびコンソールの AWS KMS カスタマーマネージドキーセクションまたは アクションを使用した KMS キーの使用状況を監査できます。 `CreateKey` AWS KMS 詳しい情報については、AWS Key Management Service デベロッパーガイドの KMS キーおよびキー作成をご参照ください。KMS キー識別子のその他の例については、 AWS Key Management Service API リファレンスKeyIdの「」を参照してください。KMS キー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**
>
> の使用には追加料金がかかります AWS KMS。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。

エンベロープ暗号化

暗号化されたデータのセキュリティは、復号できるデータキーを保護することによって部分的に異なります。Amazon SQS は KMS キーを使用してデータキーを暗号化し、暗号化されたデータキーは暗号化されたメッセージと一緒に保存されます。データキーを暗号化するために KMS キーを使用するこの方法は、エンベロープ暗号化と呼ばれます。

詳細については、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。

Amazon SQS キー管理

Amazon SQS は AWS Key Management Service (KMS) と統合して、サーバー側の暗号化 (SSE) 用の KMS キーを管理します。SSE 情報およびキー管理の定義については、「Amazon SQS での保管時の暗号化」をご参照ください。Amazon SQS は KMS キーでメッセージを暗号化および復号するデータキーを検証および保護します。以下のセクションは Amazon SQS サービスの KMS キーとデータキーの操作について説明します。

トピック

- AWS KMS 許可を設定する
- データキー再利用期間について
- AWS KMS コストの見積もり
- AWS KMS エラー

AWS KMS 許可を設定する

すべての KMS キーにはキーポリシーが必要です。Amazon SQS の AWS マネージド KMS キーのキーポリシーは変更できないことに注意してください。この KMS キーのポリシーには、暗号化されたキューを使用するためのアカウント内 (Amazon SQS の使用が承認されているもの) にあるすべてのプリンシパルのアクセス許可が含まれています。

カスタマーマネージド KMS キーの場合、各キュープロデューサーおよびコンシューマーの許可を追加するため、キーポリシーを設定する必要があります。これを行うため、KMS キーポリシー内にプロデューサーおよびコンシューマーをユーザーとして名前を付けます。アクセス AWS KMS 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 リソースとオペレーション」またはAWS KMS 「 API アクセス許可リファレンス」を参照してください。

または、必要な許可をIAM ポリシーで指定して、暗号化されたメッセージを作成および使用するプリンシパルにこのポリシーを割り当てます。詳細については、デベロッパーガイド AWS KMS の「AWS Key Management Service IAM ポリシーの使用」を参照してください。

> **Note**
>
> Amazon SQS との間で送受信するグローバル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`セクションで、特定のリージョンの KMS キーの完全な ARN に明示的に名前を付ける必要があります。

AWS サービスの KMS アクセス許可を設定する

いくつかの AWS サービスは、Amazon SQS キューにイベントを送信できるイベントソースとして機能します。これらのイベントソースが暗号化されたキューと連携できるようにするには、カスタマーマネージド KMS キーを作成し、必要な AWS KMS API メソッドを使用するためのアクセス許可をサービスのキーポリシーに追加する必要があります。アクセス許可を設定するには、次の手順を実行します。

> **Warning**
>
> Amazon SQS メッセージを暗号化するための KMS キーを変更する場合、古い KMS キーで暗号化された既存のメッセージはそのキーで暗号化されたままになることに注意してください。これらのメッセージを復号するには、古い KMS キーを保持し、そのキーポリシーが Amazon SQS に `kms:Decrypt`および のアクセス許可を付与していることを確認する必要がありま`kms:GenerateDataKey`す。新しいメッセージを暗号化するために新しい KMS キーに更新した後、古い KMS キーで暗号化された既存のすべてのメッセージが処理され、キューから削除されたことを確認してから、古い KMS キーを削除または無効にします。

1. カスタマーマネージド KMS キーを作成。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 サービスイベントソースが `kms:GenerateDataKey`および `kms:Decrypt` API メソッドを使用できるようにするには、KMS キーポリシーに次のステートメントを追加します。

```
{
   "Version": "2012-10-17",
      "Statement": [{
         "Effect": "Allow",
```

```
         "Principal": {
            "Service": "service.amazonaws.com"
         },
         "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
         ],
         "Resource": "*"
      }]
}
```

上記の例の「Service」をイベント出典のサービス名に置き換えます。イベントソースには、次のサービスが含まれます。

[イベントソース]	サービス名
Amazon CloudWatch イベント	events.amazonaws.com
Amazon S3イベント通知について	s3.amazonaws.com
Amazon SNS トピックのサブスクリプション	sns.amazonaws.com

3. KMS キーの ARN を使用して既存の SSE キューを設定します。
4. 暗号化されたキューのARNをイベント出典に追加します。

プロデューサーの AWS KMS アクセス許可を設定する

データキー再利用期間が終了した場合、プロデューサーによる次回のSendMessageまたはSendMessageBatchの呼び出しでは、kms:GenerateDataKeyとkms:Decryptの呼び出しもトリガーされます。kms:Decryptの呼び出しでは、新しいデータキーを使用する前に整合性を検証します。したがって、プロデューサーは KMS キーの kms:GenerateDataKey および kms:Decrypt 許可が必要です。

次のステートメントをプロデューサーの IAMポリシーに追加します。キーリソースとキューリソースには正しい ARN値を使用してください。

```
{
   "Version": "2012-10-17",
      "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": [
            "sqs:SendMessage"
         ],
         "Resource": "arn:aws:sqs:*:123456789012:MyQueue"
      }]
}
```

コンシューマーの AWS KMS アクセス許可を設定する

データキー再利用期間が終了した場合、コンシューマーが次回に ReceiveMessage を呼び出すと、それを使用する前に新しいデータキーの整合性を検証するために、kms:Decrypt の呼び出しもトリガーされます。したがって、コンシューマーは指定されたキューのメッセージの暗号化に使用される KMS キーへの kms:Decrypt 許可が必要です。キューがデッドレターキューとして機能する場合、コンシューマーはソースキューのメッセージの暗号化に使用される KMS キーへの kms:Decrypt 許可も必要です。次のステートメントをコンシューマーの IAM ポリシーに追加します。キーリソースとキューリソースには正しい ARN値を使用してください。

```
{
   "Version": "2012-10-17",
      "Statement": [{
         "Effect": "Allow",
         "Action": [
            "kms:Decrypt"
         ],
         "Resource": "arn:aws:kms:us-
east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
         }, {
         "Effect": "Allow",
         "Action": [
            "sqs:ReceiveMessage"
         ],
         "Resource": "arn:aws:sqs:*:123456789012:MyQueue"
      }]
```

```
}
```

混乱した代理保護によるアクセス許可の設定 AWS KMS

キーポリシーステートメントのプリンシパルが AWS サービスプリンシパルの場合、aws:SourceArn または aws:SourceAccount のグローバル条件キーを使用して混乱した使節問題から保護できます。これらの条件キーを使用する場合、暗号化されているリソースの Amazon リソースネーム (ARN) の値を設定します。リソースの ARN が不明の場合は、代わりに aws:SourceAccount を使用してください。

この KMS キーポリシーでは、アカウント 111122223333 が保有されたサービスの特定リソースは、Decrypt および GenerateDataKey アクションにおいて KMS を呼び出すことが許可されます。これは Amazon SQS の SSE 使用中に発生します。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Principal": {
            "Service": "<replaceable>service</replaceable>.amazonaws.com"
  },
  "Action": [
   "kms:GenerateDataKey",
   "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
   "ArnEquals": {
    "aws:SourceArn": [
     "arn:aws:service::111122223333:resource"
    ]
   }
  }
 }]
}
```

SSE を有効にした Amazon SQS キューを使用する場合、以下のサービスが aws:SourceArn をサポートします:

- Amazon SNS
- Amazon S3

- CloudWatch イベント
- AWS Lambda
- CodeBuild
- Amazon Connect Customer Profiles
- AWS Auto Scaling
- Amazon Chime

データキー再利用期間について

データキー再利用期間は、Amazon SQSが同じデータキーを再利用するための最大期間を定義します。データキー再利用期間が終了すると、Amazon SQSは新しいデータキーを生成します。再利用期間については、次のガイドラインに注意してください。

- 再利用期間が短いほどセキュリティが向上しますが、 への呼び出しが増え AWS KMS、 無料利用枠を超える料金が発生する可能性があります。
- データキーは暗号化用と復号化用に別々にキャッシュされますが、再利用期間はデータキーの両方のコピーに適用されます。
- データキーの再利用期間が終了すると、 への次の呼び出し、SendMessageまたはSendMessageBatch通常 メソッドへの呼び出しを AWS KMS GenerateDataKeyトリガーして、新しいデータキーを取得します。また、次に SendMessageと を呼び出すと、それぞれReceiveMessageが の AWS KMS 呼び出しをトリガーDecryptして、データキーを使用する前にデータキーの整合性を検証します。
- プリンシパル (AWS アカウント またはユーザー) はデータキーを共有しません (一意のプリンシパルによって送信されるメッセージは常に一意のデータキーを取得します) 。したがって、 への呼び出しのボリューム AWS KMS は、データキーの再利用期間中に使用されている一意のプリンシパルの数の倍数です。

AWS KMS コストの見積もり

コストを予測し、 AWS 請求をよりよく理解するために、Amazon SQS が KMS キーを使用する頻度を知りたい場合があります。

> **Note**
>
> コストは以下の計算式でかなり正確に予測できますが、Amazon SQS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) キュー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 + C)
```

B は請求期間(秒)です。

D は、データキー再利用期間(秒)です。

Pはメインの生産数は Amazon SQS キューに送信されます。

C は、Amazon SQSキューから受信する、コンシューマー側のプリンシパル数です。

> **Important**
>
> 一般的に、プロデューサー側プリンシパルで発生するコストはコンシューマー側プリンシパルの倍程度になります。詳細については、「データキー再利用期間について」を参照してください。
> プロデューサーとコンシューマーの ユーザーが異なる場合、コストは増加します。

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: 2 つのプリンシパルと 1 つのキューの AWS KMS API コール数の計算

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)に設定されています。
- キューの数は 1 つです。
- プロデューサー側プリンシパルが1つ、コンシューマー側プリンシパルが1つあります。

```
(2,678,400 / 300) * (2 * 1 + 1) = 26,784
```

例 2: 複数のプロデューサーとコンシューマー、2 つのキューに対する AWS KMS API コールの数を計算する

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24時間(86,400秒)に設定されています。
- キューの数は2つです。
- 1つ目のキューのプロデューサー側プリンシパルは3つ、コンシューマー側プリンシパルは1つです。
- 2つ目のキューのプロデューサー側プリンシパルは5つ、コンシューマー側プリンシパルは2つです。

```
(2,419,200 / 86,400 * (2 * 3 + 1)) + (2,419,200 / 86,400 * (2 * 5 + 2)) = 532
```

AWS KMS エラー

Amazon SQS と を操作すると AWS KMS、エラーが発生する可能性があります。次のリファレンスでは、エラーおよび考えられるトラブルシューティング方法について説明します。

- AWS KMS の一般的なエラー
- AWS KMS 復号エラー
- AWS KMS GenerateDataKey エラー

Amazon SQS でのネットワークトラフィックのプライバシー

Amazon SQSの Amazon Virtual Private Cloud (Amazon VPC) エンドポイントは、Amazon SQS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 SQS 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。

トピック

- Amazon SQSのAmazon Virtual Private Cloud エンドポイント
- Amazon SQS用のAmazon VPCエンドポイントポリシーを作成する

Amazon SQSのAmazon Virtual Private Cloud エンドポイント

Amazon VPC を使用して AWS リソースをホストする場合、VPC と Amazon SQS 間の接続を確立できます。この接続を使用して、公開インターネットと交差せずにAmazon SQSキューにメッセージを送信できます。

Amazon VPC では、カスタム仮想ネットワークで AWS リソースを起動できます。VPCを使用して、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。Amazon VPCの詳細については、Amazon VPC ユーザーガイドを参照してください。

VPCを Amazon SQSに接続するには、最初にインターフェイス VPC エンドポイントを使用すると、VPCを他の AWS のサービス VPCに接続できます。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 SQS 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「チュートリアル:Amazon Virtual Private Cloud から Amazon SQSキューにメッセージを送信するそして例5:VPC エンドポイントからではない場合はアクセスを拒否するこのガイドのインターフェイス VPC エンドポイント (AWS PrivateLink)の」を参照してください。

> **Important**
>
> - Amazon Virtual Private Cloud は HTTPS Amazon SQS エンドポイントでのみ使用できます。
> - Amazon VPCからメッセージを送信するように Amazon SQS を設定する場合、プライベート DNS を有効にして、`sqs.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。

Amazon SQS用のAmazon VPCエンドポイントポリシーを作成する

Amazon SQSのAmazon VPC エンドポイントに対するポリシーを作成して、以下を特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 「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

次の VPC エンドポイントポリシーの例では、Amazon SQS キュー MyQueue にメッセージを送信することを MyUser ユーザーに許可するように指定しています。

```
{
   "Statement": [{
      "Action": ["sqs:SendMessage"],
      "Effect": "Allow",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Principal": {
        "AWS": "arn:aws:iam:123456789012:user/MyUser"
      }
   }]
}
```

以下は拒否されます:

- sqs:CreateQueueやsqs:DeleteQueueなど他の Amazon SQS API アクション
- この VPC エンドポイントを使用しようとする の他のユーザーおよびルール。
- MyUser別のAmazon SQS キューにメッセージを送信する。

> **Note**
>
> ユーザーはさらに、他の Amazon SQS API アクションを VPC の外側からでも使用できます。詳細については、「例5:VPC エンドポイントからではない場合はアクセスを拒否する」を参照してください。

Amazon SQSでの Identity and Access Management

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 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインインを許可) し、誰に Amazon SQS 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、追加料金なしで AWS のサービス 使用できる です。

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon SQS で行う作業によって異なります。

サービスユーザー – 業務を行うために Amazon SQS サービスを使用する場合、管理者が必要な認証情報および許可を提供します。業務のために使用する Amazon SQS 機能が増えるにつれ、追加の許可が必要な場合があります。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 SQS の機能にアクセスできない場合は、「Amazon Simple Queue 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 SQS リソースを担当している場合は、通常、Amazon SQS 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 SQS 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 SQS と IAM を併用する方法の詳細については、「Amazon Simple Queue 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 SQS 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 SQS 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、「ポリシーのベストプラクティス」を参照してください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID 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 AWS を使用して にアクセスすると、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。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用して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。推奨される方法を使用してリクエストに自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS 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、追加セキュリティ情報の提供をリクエストさ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向上させることをお勧めします。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 ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての AWS のサービス およびリソースへの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 ID は AWS アカウント 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でアクセスできます。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は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。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アイデンティティ

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、一時的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービス します。

フェデレーティッド ID 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、または ID ソースを通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID が にアクセスすると AWS アカウント、ロールを引き受け、ロールは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 Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「IAM Identity Center 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、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 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許可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、特定のアクセス許可 AWS アカウント を持つ 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 AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、 または AWS API AWS CLI 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、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID 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 ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。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「Creating a role for a third-party Identity Provider」(サードパーティーアイデンティティプロバイダー向けロールの作成) を参照してください。IAM Identity Center を使用する場合は、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 Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」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にアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) 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 では AWS のサービス、 (ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスの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 でのクロスアカウントリソースアクセス」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。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許可、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 でアクションを実行する場合 AWS、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、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。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 のサービスにアクセス許可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、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。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、インスタンスにアタッチ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の管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 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) AWS がリクエストを行うときに、これらのポリシー AWS 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。ほとんどのポリシーは JSON ドキュメント AWS として 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 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。

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、アクションの許可を定義します。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。そのポリシーを持つユーザーは、 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のポリシー

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、および条件をコントロールします。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ています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。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には例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。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、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

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を使用できません。

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、ACLs。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 は、一般的ではない追加のポリシータイプをサポートします。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た最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、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) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可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アクセス許可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、アクセス許可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。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するためのサービスです。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 は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。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ション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」を参照してください。

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与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

Amazon SQS でのアクセス管理の概要

すべての AWS リソースは によって所有され AWS アカウント、リソースを作成またはアクセスするためのアクセス許可はアクセス許可ポリシーによって管理されます。アカウント管理者は、IAMアクセス権限ポリシーをアイデンティティ (ユーザー、グループ、ロール) にアタッチできます。一部のサービス (Amazon SQS など) では、アクセス権限ポリシーをリソースにアタッチすることもできます。

> **Note**
>
> アカウント管理者 (または管理者ユーザー) は、管理者権限を持つユーザーです。詳細については、「IAM ユーザーガイド」の「IAM のベストプラクティス」を参照してください。

アクセス権限を付与する場合、アクセス権限を取得するユーザー、取得するアクセス権限の対象となるリソース、およびリソースに対して許可される特定のアクションを指定します。

トピック

- Amazon Simpleキューサービス リソースと操作
- リソース所有権について
- リソースへのアクセスの管理
- ポリシー要素の指定:アクション、効果、リソース、プリンシパル

Amazon Simpleキューサービス リソースと操作

Amazon SQS では、唯一のリソースはキューです。ポリシーで Amazon リソースネーム (ARN)を使用して、ポリシーを適用するリソースを識別します。次のリソースには、関連付けられた一意の ARN があります。

リソースタイプ	ARN 形式
キュー	`arn:aws:sqs: region:account_id :queue_name`

キューの ARN 形式の例を以下に示します。

- AWS アカウント 123456789012 に属する、米国東部 (オハイオ) リージョンmy_queueの という名前のキューの ARN。

```
arn:aws:sqs:us-east-2:123456789012:my_queue
```

- Amazon SQSがサポートする異なるリージョンごとのmy_queueというキューのARN。

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

- キュー名に対して*または?をワイルドカードとして使用する ARN。次の例で、ARN はプレフィックス my_prefix_ が付いたすべてのキューに一致します。

```
arn:aws:sqs:*:123456789012:my_prefix_*
```

GetQueueAttributes アクションを呼び出して既存のキューの ARN 値を取得できます。QueueArn 属性の値は、キューの ARN です。ARNの詳細については、IAM ユーザーガイドの「IAM ARN」を参照してください。

Amazon SQSには、キューリソースを操作するための一連の アクションが用意されています。詳細については、「Amazon SQS (Amazon SQS)、 API のアクセス権限: アクションとリソースのリファレンスについて」を参照してください。

リソース所有権について

は、リソースを作成したユーザーに関係なく、アカウントで作成されたリソース AWS アカウントを所有します。具体的には、リソース所有者は、リソースの作成リクエストを認証するプリンシパルエンティティ (ルートアカウント、ユーザー、または IAM ロール) の AWS アカウント です。次の例は、この仕組みを示しています。

- のルートアカウントの認証情報を使用して Amazon SQS キュー AWS アカウント を作成する場合、 AWS アカウント はリソースの所有者です (Amazon SQS では、リソースは Amazon SQS キューです) 。
- でユーザーを作成し AWS アカウント 、そのユーザーにキューを作成するアクセス許可を付与すると、そのユーザーはキューを作成できます。ただし、(ユーザーが属する) AWS アカウント アカウントはキューリソースを所有しています。
- Amazon SQS キューを作成するアクセス許可 AWS アカウント を持つ で IAM ロールを作成する場合、ロールを引き受けることのできるいずれのユーザーもキューを作成できます。 Amazon SQS （ ロールが属 AWS アカウント する) がキューリソースを所有します。

リソースへのアクセスの管理

アクセス許可ポリシー では、誰が何にアクセスできるかを記述します。以下のセクションで、アクセス権限のポリシーを作成するために使用可能なオプションについて説明します。

> **Note**
>
> このセクションでは、Amazon SQSのコンテキストでの IAM の使用について説明します。これは、IAMサービスに関する詳細情報を取得できません。IAM に関する詳細なドキュメントについては、「IAM ユーザーガイド」の「What is IAM?」(IAM とは？) を参照してください。IAM ポリシーの構文と説明については、「IAM ユーザーガイド」の「AWS IAM ポリシーリファレンス」を参照してください。

IAM ID にアタッチされたポリシーは ID ベースのポリシー (IAM ポリシー) と呼ばれ、リソースにアタッチされたポリシーはリソースベースのポリシーと呼ばれます。

アイデンティティベースのポリシー

ユーザーにAmazon SQSキューのアクセス権限を付与する方法は、 ポリシーシステムを使用する方法とIAMポリシーシステムを使用する方法の 2 つです。いずれかのシステムまたは両方を使用して、ユーザーまたはロールにポリシーをアタッチできます。ほとんどの場合、どちらのシステムを使用しても同じ結果が得られます。例えば、次の操作を実行できます:

- アカウントのユーザーまたはグループに許可ポリシーをアタッチする–Amazon SQS キューを作成する許可を付与するために、ユーザーまたはユーザーが所属するグループに許可ポリシーをアタッチできます。
- 別の AWS アカウントアカウントのユーザーにアクセス権限ポリシーをアタッチする- ユーザーにAmazon SQSキューを作成するアクセス権限を付与するには、別の AWS アカウントにAmazon SQSアクセス権限をアタッチします。

  クロスアカウントアクセス権限は、次のアクションには適用されません。

  - `AddPermission`
  - `CancelMessageMoveTask`
  - `CreateQueue`
  - `DeleteQueue`
  - `ListMessageMoveTask`
  - `ListQueues`
  - `ListQueueTags`
  - `RemovePermission`
  - `SetQueueAttributes`

  - StartMessageMoveTask
  - TagQueue
  - UntagQueue
- アクセス権限ポリシーをロールにアタッチする (クロスアカウントのアクセス権限を付与する) –クロスアカウントのアクセス権限を付与するには、アイデンティティベースのアクセス権限ポリシーを IAM ロールにアタッチできます。例えば、 AWS アカウント A 管理者は、次のように AWS アカウント B (または AWS サービス) にクロスアカウントアクセス許可を付与するロールを作成できます。
  - アカウント A の管理者は、IAM ロールを作成し、そのロールに対して、アカウント A のリソースにアクセス許可を付与するアクセス許可ポリシーをアタッチします。
  - アカウント Aの管理者は、アカウントBをそのロールを引き受けるプリンシパルとして識別するロールに、信頼ポリシーを添付します。
  - アカウントBの管理者は、ロールを引き受けるアクセス権限をアカウントBの任意のユーザーに委任できます。これにより、アカウントBのユーザーはアカウントAでキューを作成したり、キューにアクセスしたりできます。

    > **Note**
    >
    > ロールを引き受けるアクセス許可を AWS サービスに付与する場合、信頼ポリシーのプリンシパルは AWS サービスプリンシパルにすることもできます。

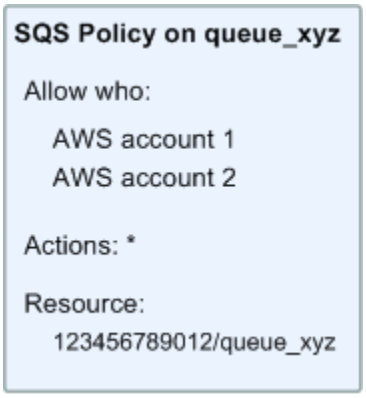
IAM を使用した許可の委任の詳細については、「IAM ユーザーガイド」の「アクセス管理」を参照してください。

Amazon SQS はIAMポリシーを使用して作業する一方で、独自のポリシーインフラストラクチャがあります。Amazon SQS ポリシーをキューとともに使用して、キューにアクセスできる AWS アカウントを指定できます。アクセスタイプと条件を指定できます (たとえば、リクエストが 2010年12月31日より前の場合はSendMessage、ReceiveMessageを使用するアクセス権限を付与する条件)。アクセス許可を付与できる特定のアクションは、Amazon SQS アクションリスト全体のサブセットです。Amazon SQS ポリシーを記述して、*「すべての アクションを許可」を意味するを指定した場合、ユーザーがこのサブセット内のすべてのアクション実行できることを意味します。

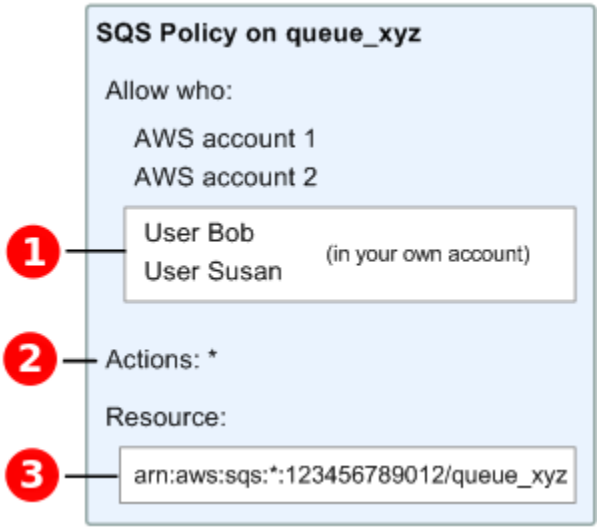
次の図は、これらのベーシックなAmazon SQS ポリシーのうち1つの概念を表しており、アクションのサブセットを取り上げています。ポリシーは 用でありqueue_xyz、 AWS アカウント 1 と AWS

アカウント 2 に、指定されたキューで許可されたアクションのいずれかを使用するアクセス許可を付与します。

> **Note**
>
> ポリシーのリソースは として指定されます。ここで123456789012/queue_xyz、123456789012はキューを所有する AWS アカウントのアカウント ID です。



IAMと、ユーザーおよび Amazon リソースネーム (ARN)の概念の導入により、SQSポリシーに関するいくつかの点が変わりました。次の図と表は、その変更を示しています。



1

異なるアカウントのユーザーにアクセス許可を付与する方法については、IAM ユーザーガイドの「チュートリアル: IAM ロールを使用した AWS アカウント間のアクセスの委任」を参照してください。

2

*に含まれるアクションのサブセットが拡大されました。可能なアクションの一覧については、Amazon SQS (Amazon SQS)、 API のアクセス権限: アクションとリソースのリファレンスについてを参照してください。

3

Amazo

リソースネーム (ARN)を使用してリソースを指定できます。これは、IAMポリシーでリソースを指定するためのスタンダードな方法です。Amazon SQSキューのARN形式については、「Amazon Simpleキューサービス リソースと操作」を参照してください。

例えば、前の図の Amazon SQS ポリシーによると、 AWS アカウント 1 または AWS アカウント 2 のセキュリティ認証情報を所有するすべてのユーザーが にアクセスできますqueue_xyz。さらに、自身の AWS アカウント (ID 123456789012)内のユーザー Bobおよび Susanもキューにアクセスできます。

IAMの導入前は、 Amazon SQSによりキューの作成者に、キューに対する完全なコントロールが付与されていました (そのキューで使用できるすべての Amazon SQ アクションへのアクセス)。これは、作成者が AWS セキュリティ認証情報を使用している場合以外は当てはまらなくなりました。キューを作成するアクセス権限を持つユーザーは、作成されたキューで何らかの操作を実行するには、他のAmazon SQSアクションを使用するアクセス権限も持っている必要があります。

以下に、ユーザーにすべてのAmazon SQSアクションを許可するが、対象を名前にリテラル文字列というプレフィックスがついているキューに限るポリシーの例を示しますbob_queue_。

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

詳細については、IAMユーザーガイドAmazon SQS でのポリシーの使用「ID (ユーザー、グループ、ロール)」とのIAM ポリシーの概要」を参照してください。

ポリシー要素の指定:アクション、効果、リソース、プリンシパル

Amazon Simple キューサービス リソースの種類ごとに、このサービスは、一連のアクションを定義します。これらの アクションを実行するためのアクセス権限を付与するために、Amazon SQSではポリシーに一連のアクションを定義できます。

> **Note**
>
> 1つの アクションの実行で、複数のアクションのアクセス権限が必要になる場合があります。特定のアクションのアクセス権限を付与した場合は、アクションを許可または拒否するリソースを識別します。

最も基本的なポリシーの要素を次に示します。

- リソース–ポリシーで Amazon リソースネーム (ARN)を使用して、ポリシーを適用するリソースを識別します。
- アクション–アクションのキーワードを使用して、許可または拒否するリソースアクションを識別します。たとえば、`sqs:CreateQueue`権限は、Amazon Simple キューサービス`CreateQueue`アクション の実行をユーザーに許可します。
- 効果–ユーザーが特定のアクションを要求する際の効果を指定します。許可または拒否のいずれかになります。リソースへのアクセスを明示的に許可していない場合、アクセスは暗黙的に拒否されます。また、明示的にリソースへのアクセスを拒否すると、別のポリシーによってアクセスが許可されている場合でも、ユーザーはそのリソースにアクセスできなくなります。
- プリンシパル-アイデンティティベースのポリシー (IAMポリシー) で、ポリシーが添付されているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、権限 (リソースベースのポリシーにのみ適用) を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。

Amazon SQS ポリシーの構文と記述の詳細については、IAM ユーザーガイドのAWS IAM ポリシーリファレンスを参照してください。

すべてのAmazon Simpleキューサービスアクションおよびそれに適用されるリソースを示す表については、「Amazon SQS (Amazon SQS)、 API のアクセス権限: アクションとリソースのリファレンスについて」を参照してください。

Amazon Simple Queue Service で IAM を使用する方法

IAM を使用して Amazon SQS へのアクセスを管理する前に、Amazon SQS で利用できる IAM 機能について理解しておく必要があります。

Amazon Simple Queue Service で使用できる IAM の機能

IAM 機能	Amazon SQS サポート
アイデンティティベースのポリシー	Yes
リソースベースのポリシー	はい
ポリシーアクション	Yes
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	No
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	はい
転送アクセスセッション (FAS)	はい
サービスロール	あり
サービスリンクロール	いいえ

Amazon SQS およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

アクセスコントロール

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コントロールします。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、ACLs。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。

> **Note**
>
> すべての が自分のアカウントでアクセス許可をユーザーに委任 AWS アカウント 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を使用すると、追加のユーザーを管理することなく、 AWS リソースへのアクセスを共有できます。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 SQS カスタムポリシー内のコンテンツ間のアクセス許可と条件キーの詳細については、「Amazon SQS カスタムポリシーの制限事項」を参照してください。

Amazon SQS 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	Yes

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 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、および条件をコントロールします。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、それが添付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 SQS のアイデンティティベースのポリシー例

Amazon SQS のアイデンティティベースのポリシー例を確認するには、「ポリシーのベストプラクティス」を参照してください。

Amazon SQS 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には例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。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、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、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 にある場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。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さらに付与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 でのクロスアカウントリソースアクセス」を参照してください。

Amazon SQS のポリシーアクション

ポリシーアクションのサポート	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。

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、関連付けられた 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許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数の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 SQS 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「Amazon Simple Queue Service で定義されるリソース」を参照してください。

Amazon SQS 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。

```
sqs
```

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": [
      "sqs:action1",
      "sqs:action2"
         ]
```

Amazon SQS のアイデンティティベースのポリシー例を確認するには、「ポリシーのベストプラクティス」を参照してください。

Amazon SQS 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	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のリソースにどのよう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 ポリシー要素は、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、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、リソースレベルの許可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 SQS の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「Amazon Simple Queue 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、「Amazon Simple Queue Service で定義されるリソース」を参照してください。

Amazon SQS のアイデンティティベースのポリシー例を確認するには、「ポリシーのベストプラクティス」を参照してください。

Amazon SQS 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。イコールや未満などの 条件演算子 を使用して条件式を作成することで、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` 要素を指定するか、1 つの `Condition` 要素に複数のキーを指定すると、 AWS は `AND` 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理`OR`オペレーションを使用して条件 AWS 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権限を付与することができます。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 SQS 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「Amazon Simple Queue 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、「Amazon Simple Queue Service で定義されるリソース」を参照してください。

Amazon SQS のアイデンティティベースのポリシー例を確認するには、「ポリシーのベストプラクティス」を参照してください。

Amazon SQS での ACL

ACL のサポート	No

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コントロールします。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 SQS 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制御 (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。では AWS、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) および多くの 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。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

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管理するには、`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はいです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

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「属性ベースのアクセス制御 (ABAC) を使用する」を参照してください。

Amazon SQS 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	はい

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、IAM ユーザーガイドのAWS のサービス「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社の Single Sign-On (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。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。その後、これらの一時的な認証情報を使用して . AWS recommends にアクセスできます AWS。これは、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推奨しています。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

Amazon SQS の転送アクセスセッション

転送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 でアクションを実行すると AWS、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、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。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 SQS のサービスロール

サービスロールのサポート	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 のサービスにアクセス許可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 SQS の機能が破損する可能性があります。Amazon SQS が指示するときにのみ、サービスロールを編集してください。

Amazon SQS のサービスリンクロール

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、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、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権限を表示できますが、編集することはできません。

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、Service-linked role (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。サービスリンク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。

AWS マネージドポリシーに対する Amazon SQS の更新

ユーザー、グループ、ロールにアクセス許可を追加するには、自分でポリシーを作成するよりも、AWS 管理ポリシーを使用する方が簡単です。チームに必要な権限のみを提供する IAM カスタマーマネージドポリシーを作成するには、時間と専門知識が必要です。すぐに使用を開始するために、AWS マネージドポリシーを使用できます。これらのポリシーは、一般的なユースケースを対象範囲に含めており、 AWS アカウントで利用できます。 AWS 管理ポリシーの詳細については、「IAM ユーザーガイド」の「 AWS 管理ポリシー」を参照してください。

AWS サービスは、 AWS マネージドポリシーを維持および更新します。 AWS 管理ポリシーのアクセス許可は変更できません。サービスは、新機能をサポートするために、 AWS マネージドポリシー

に追加のアクセス許可を追加することがあります。この種の更新は、ポリシーがアタッチされているすべてのアイデンティティ (ユーザー、グループ、ロール) に影響を与えます。サービスは、新機能の起動時または新しいオペレーションが利用可能になったときに、 AWS マネージドポリシーを更新する可能性が最も高くなります。サービスは AWS 管理ポリシーからアクセス許可を削除しないため、ポリシーの更新によって既存のアクセス許可が破損することはありません。

さらに、 は、複数の サービスにまたがる職務機能の マネージドポリシー AWS をサポートします。例えば、 ReadOnlyアクセス AWS 管理ポリシーは、すべての AWS サービスとリソースへの読み取り専用アクセスを提供します。サービスが新機能を起動すると、 は新しいオペレーションとリソースの読み取り専用アクセス許可 AWS を追加します。ジョブ機能のポリシーの一覧および詳細については、「IAM ユーザーガイド」の「AWS のジョブ機能のマネージドポリシー」を参照してください。

AWS マネージドポリシー: AmazonSQSFullAccess

`AmazonSQSFullAccess` ポリシーは Amazon SQS ID に添付できます。このポリシーは、Amazon SQS への完全なアクセスを可能にする許可を付与します。

このポリシーのアクセス許可を確認するには、「 マネージドポリシーリファレンス」のAmazonSQSFullAccess」を参照してください。 AWS

AWS マネージドポリシー: AmazonSQSReadOnly アクセス

`AmazonSQSReadOnlyAccess` ポリシーは Amazon SQS ID にアタッチできます。このポリシーは、Amazon SQS への読み取り専用アクセスを可能にする許可を付与します。

このポリシーのアクセス許可を確認するには、「 マネージドポリシーリファレンス」のAmazonSQSReadOnly アクセス」を参照してください。 AWS

AWS マネージドポリシーに対する Amazon SQS の更新

Amazon SQS の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。このページへの変更に関する自動アラートを受け取るには、Amazon SQS のドキュメント履歴ページで RSS フィードにサブスクライブしてください。

変更	説明	日付
AmazonSQSReadOnly アクセス	Amazon SQS には、特定のソースキューにある最新のメッセージ移動タスク (最大	2023 年 6 月 9 日

変更	説明	日付
	10 個) を一覧表示できる新しいアクションが追加されました。このアクションは ListMessageMoveTasks API 演算に関連付けられています。	

Amazon Simple Queue 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 SQS 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 SQS 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 以外のユーザーに Amazon SQS リソース AWS アカウント へのアクセスを許可したい

Amazon SQS 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 sqs:*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:
 sqs: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 sqs:*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 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

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 SQS 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またはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを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 SQS 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。メアリーには、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メアリーに `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 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

自分の 以外のユーザーに Amazon SQS リソース AWS アカウント 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コントロールリスト (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 SQS 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 Queue Service で IAM を使用する方法」を参照してください。
- 所有 AWS アカウント している のリソースへの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「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。

- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、「IAM ユーザーガイド」の「サードパーティー AWS アカウント が所有する へのアクセスを提供する」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認証されたユーザーへのアクセスの提供 (ID フェデレーション) 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については、IAM ユーザーガイドの「IAM でのクロスアカウントリソースアクセス」を参照してください。

Amazon SQS でのポリシーの使用

このトピックでは、アカウント管理者がIAMアイデンティティ (ユーザー、グループ、ロール) へのアクセス権限ポリシーをアタッチする、アイデンティティベースのポリシーの例を示します。

> **Important**
>
> 初めに Amazon Simpleキューサ - ビスリソースへのアクセスを管理するための基本概念と使用できるオプションについて説明する概要トピックをご覧になることをお勧めします。詳細については、「Amazon SQS でのアクセス管理の概要」を参照してください。
> ただし、`ListQueues`では、すべての Amazon SQSアクションがリソース-レベルのアクセス許可をサポートします。詳細については、「Amazon SQS (Amazon SQS)、 API のアクセス権限: アクションとリソースのリファレンスについて」を参照してください。

トピック

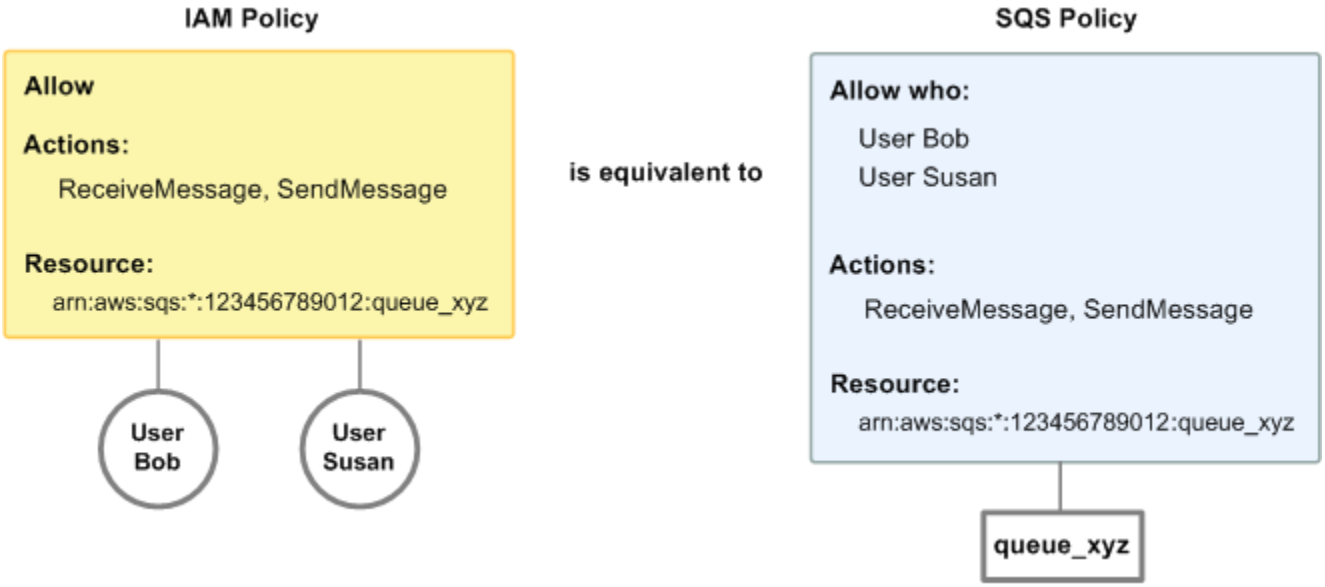
- Amazon SQS ポリシーとIAMポリシーの使用
- Amazon SQSコンソールの使用に必要なアクセス許可
- Amazon SQS のアイデンティティベースのポリシー例
- ベーシックAmazon SQSポリシーの例
- Amazon SQSアクセスポリシー言語を使用したカスタムポリシーの使用

Amazon SQS ポリシーとIAMポリシーの使用

ユーザーに Amazon SQSリソースのアクセス権限を付与する方法は、 ポリシーシステムを使用する方法とIAMポリシーシステムを使用する方法の 2 つです。いずれか、または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、IAMポリシーとそれに相当する Amazon SQS ポリシーを示しています。IAM ポリシーは、 AWS アカウントqueue_xyzで というキューの Amazon SQS ReceiveMessageおよび SendMessageアクションに対する権限を付与し、ポリシーは Bob と Susan という名前のユーザーにアタッチされます (Bob と Susan にはポリシーに記載されているアクセス許可があります) 。この ReceiveMessageAmazon SQS ポリシーは、Bob と Susan に、同じキューの と SendMessage アクションへの権限も付与します。

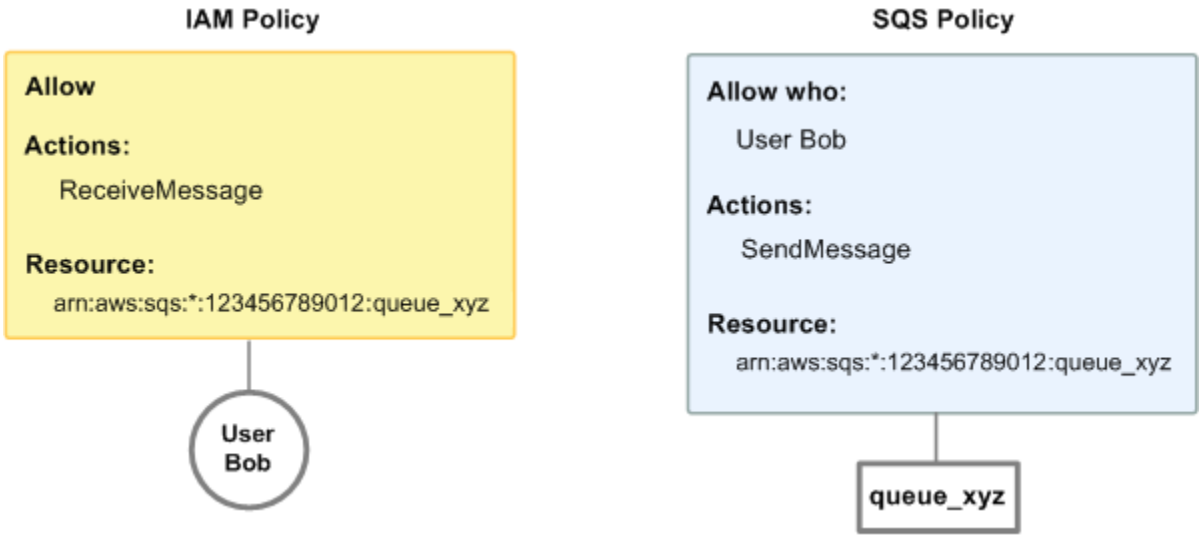
> **Note**
>
> 次の例は、条件のない単純なポリシーを示しています。どちらのポリシーでも特定の条件を指定して、同じ結果を得ることができます。



IAM ポリシーと Amazon SQS ポリシーには大きな違いが 1 つあります。Amazon SQS 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 では付与されません。

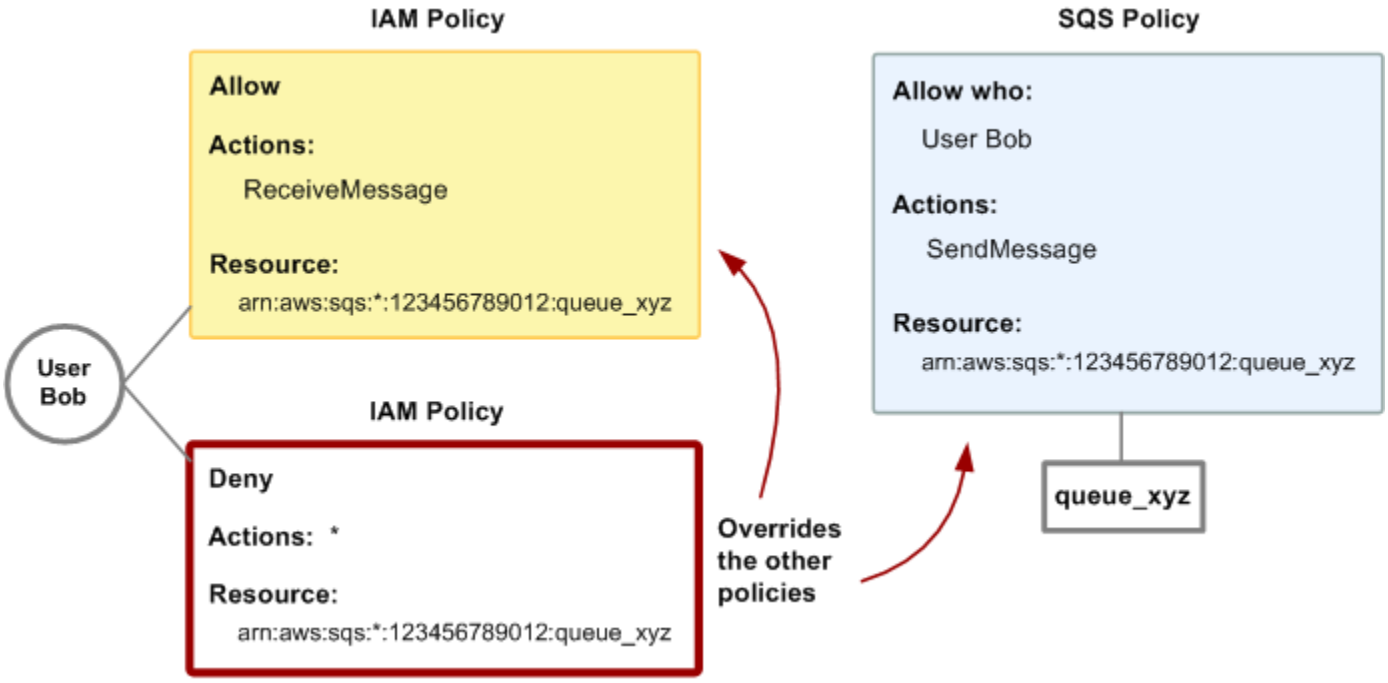
両方のシステムを同時に使用してどのようにアクセス許可を管理するかは、ご自身で決めてください。以下の例では、2つのポリシーシステムがどのように連携するかを示しています。

- 最初の例では、BobのアカウントにIAMポリシーとAmazon SQS ポリシーの両方が適用されています。ReceiveMessageIAMポリシーは、Amazon SQSでのqueue_xyzアクションについて、Bob のアカウントにアクセス許可を付与しAmazon SQS 、 ポリシーは、同じキューのSendMessageアクションのアクセス許可をBobのアカウントに付与します。以下の図に、そのコンセプトを示します。



Bob がReceiveMessageリクエストをqueue_xyzに送信すると、IAMポリシーでそのアクションが許可されます。BobがSendMessageリクエストをqueue_xyz に送信すると Amazon SQSポリシーでそのアクションが許可されます。

- 2 番目の例では、Bob がqueue_xyzに対するアクセス許可を不正に使用したため、このキューへのBobのアクセス許可をすべて削除する必要があります。最も簡単な方法は、そのキューに対するBobのアクションをすべて拒否するようなポリシーを追加することです。明示的な denyは常にallowをオーバーライドするため、このポリシーは他の 2 つのポリシーをオーバーライドします。ポリシーの評価ロジックの詳細については、「Amazon SQSアクセスポリシー言語を使用したカスタムポリシーの使用」を参照してください。以下の図に、そのコンセプトを示します。



また、そのキューに対する Bob からのアクセスをすべて拒否するようなステートメントを Amazon SQSポリシーに追加することもできます。これは、そのキューに対する Bobのアクセスを拒否するIAMポリシーを追加するのと同じ効果を持ちます。Amazon SQS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、「ベーシックAmazon SQSポリシーの例」を参照してください。AmazonSQS記載 ポリシーの詳細については、「Amazon SQSアクセスポリシー言語を使用したカスタムポリシーの使用」を参照してください。

Amazon SQSコンソールの使用に必要なアクセス許可

Amazon SQSコンソールを使用して作業するユーザーには、ユーザーの AWS アカウントアカウントのAmazon SQSキューを使用するための最小限のアクセス権限が必要です。たとえば、ユーザーにはキューをリスト表示するための `ListQueues` アクションを呼び出すアクセス権限、またはキューを作成するための `CreateQueue` アクションを呼び出すアクセス権限が必要です。Amazon SQS キューを Amazon SNSトピックに登録する Amazon SQS アクセス権限に加えて、コンソールには Amazon SNSアクション用のアクセス権限が必要です。

これらの最小限必要なアクセス権限よりも制限された IAMポリシーを作成している場合、そのIAMポリシーを使用するユーザーに対してコンソールは意図したとおりには機能しない場合があります。

AWS CLI または Amazon SQS アクションのみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。

Amazon SQS のアイデンティティベースのポリシー例

デフォルトでは、ユーザーとロールには Amazon SQS リソースを作成または変更するアクセス許可がありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。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 SQS 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「Amazon Simple Queue 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。

> **Note**
>
> Amazon EC2 Auto Scaling のライフサイクルフックを構成する場合、メッセージをAmazon SQSキューに送信するためのポリシーを作成する必要はありません。詳細については、Amazon EC2 ユーザーガイド」の「Amazon EC2 Auto Scaling ライフサイクルフック」を参照してください。 Amazon EC2

トピック

- ポリシーのベストプラクティス
- Amazon SQS コンソールの使用
- 自分の権限の表示をユーザーに許可する
- ユーザーにキューの作成を許可する
- デベロッパーが共有キューにメッセージを書き込むことを許可する
- マネージャーがキューの一般的なサイズを取得できるようにする
- パートナーが特定のキューにメッセージを送信することを許可する

ポリシーのベストプラクティス

アイデンティティベースのポリシーは、誰がユーザーのアカウントの Amazon SQS リソースを作成、アクセス、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 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可を付与するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義して、アクセス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場合は、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、最小特権アクセス許可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 でのポリシーとアクセス許可」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。条件を使用して、 などの特定の を介してサービスアクションが使用される場合に AWS のサービス、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 AWS CloudFormation。詳細については、「IAM ユーザーガイド」の [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素: 条件) を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer 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 Access Analyzer 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 – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。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 SQS コンソールの使用

Amazon Simple Queue Service コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 の Amazon SQS リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な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) に対して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アクセス許可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が許可されます。

ユーザーとロールが引き続き Amazon SQS 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 SQS `AmazonSQSReadOnlyAccess` 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「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

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 または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

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

ユーザーにキューの作成を許可する

次の例では、すべてのAmazon SQSアクションへのアクセスを Bobに許可するBob用のポリシーを作成しますが、名前のプレフィックスがリテラル文字列 alice_queue_ になっているキューしか含まれていません。

Amazon SQSでは、キューの作成者に、そのキューを使用するアクセス権限が自動的に付与されません。したがって、IAM ポリシーの CreateQueue アクションに加えて、すべての Amazon SQS アクションを使用するアクセス権限を Bob に明示的に付与する必要があります。

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

デベロッパーが共有キューにメッセージを書き込むことを許可する

次の例では、デベロッパー用のグループを作成し、グループが Amazon SQS SendMessageアクションを使用できるようにするポリシーをアタッチしますが、指定された に属 AWS アカウントし、 という名前のキューでのみ使用しますMyCompanyQueue。

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

* の代わりにSendMessageを使用してアクション ChangeMessageVisibility、DeleteMessage、GetQueueAttributes、GetQueueUrl、ReceiveMe およびSendMessageを共有キューのプリンシパルに許可できます。

> **Note**
>
> *には、他のアクセス許可タイプにより提供されるアクセスが含まれています、Amazon SQS はアクセス許可を個別に考慮します。たとえば、* により許可されるアクセスがSendMessageに含まれている場合でも、*とSendMessageの両方のアクセス許可をユーザーに付与できます。
> このコンセプトは、アクセス許可を削除するときも適用されます。プリンシパルに*アクセス許可しかない場合は、SendMessageアクセス許可の削除をリクエストしても、プリンシパルは「everything-but」アクセス許可を持つことにはなりません。プリンシパルは明示的なSendMessageアクセス許可を持っていないため、リクエストに効果はありません。プリンシパルがReceiveMessageアクセス許可だけを持つようにする場合、まずReceiveMessage アクセス許可を追加してから*アクセス許可を削除します。

マネージャーがキューの一般的なサイズを取得できるようにする

次の例では、マネージャー用のグループを作成し、指定された AWS アカウントに属するすべてのキューで Amazon SQS GetQueueAttributesアクションをグループが使用できるようにするポリシーをアタッチします。

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

パートナーが特定のキューにメッセージを送信することを許可する

このタスクは、Amazon SQSポリシーまたはIAMポリシーを使用して完了できます。パートナーにがある場合は AWS アカウント、Amazon SQS ポリシーを使用する方が簡単です。ただし、 AWS セキュリティ認証情報を所有するパートナーの会社のユーザーは、キューにメッセージを送信できます。特定のユーザーまたはアプリケーションにアクセスを制限する場合は、パートナーを自社内のユーザーと同様に扱い、IAMポリシーではなく Amazon SQSポリシーを使用する必要があります。

この例は以下のアクションを実行します。

1. パートナー会社を表す WidgetCo というグループを作成します。

2. アクセスを必要としているパートナー会社の特定のユーザーまたはアプリケーション用のユーザーを作成します。
3. ユーザーを グループに追加します。
4. SendMessageというキューのみのWidgetPartnerQueueアクションのみへのグループのアクセス権限を付与するポリシーを添付します。

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

ベーシックAmazon SQSポリシーの例

このセクションでは、一般的なAmazon SQSユースケースのサンプルポリシーを示します。

コンソールを使用して、ユーザーにポリシーをアタッチしながら各ポリシーの効果を検証できます。最初は、ユーザーにアクセス権限が付与されてないため、コンソールを使用して何もできません。ユーザーにポリシーをアタッチすることで、ユーザーがコンソールで多様なアクションを実行できることを確認できます。

> **Note**
>
> 2 つのブラウザウィンドウを使用することをお勧めします。1 つはアクセス許可を付与し、もう 1 つはユーザーの認証情報 AWS Management Console を使用して にサインインし、アクセス許可をユーザーに付与するときに確認することです。

例 1: 1 つのアクセス許可を 1 つに付与する AWS アカウント

次のポリシー例では111122223333、米国東部 (オハイオ) リージョンで という名前のキューのSendMessageアクセス許可を AWS アカウント 番号444455556666/queue1に付与します。

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

```
   "Id": "Queue1_Policy_UUID",
   "Statement": [{
      "Sid":"Queue1_SendMessage",
      "Effect": "Allow",
      "Principal": {
         "AWS": [
            "111122223333"
         ]
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:us-east-2:444455556666:queue1"
   }]
}
```

例 2: 1 つの に 2 つのアクセス許可を付与する AWS アカウント

次のポリシー例では、 という名前のキューの SendMessageおよび アクセスReceiveMessage許可111122223333の両方を AWS アカウント 番号に付与します444455556666/queue1。

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

例 3: 2 つの にすべてのアクセス許可を付与する AWS アカウント

次のポリシー例では、米国東部 (111122223333オハイオ444455556666) リージョン123456789012/queue1で という名前のキューの共有アクセスを Amazon SQS が許可するすべ

てのアクションを使用するための 2 つの異なる AWS アカウント 番号 (および) のアクセス許可を付与します。

```
{
   "Version": "2012-10-17",
   "Id": "Queue1_Policy_UUID",
   "Statement": [{
      "Sid":"Queue1_AllActions",
      "Effect": "Allow",
      "Principal": {
         "AWS": [
            "111122223333",
            "444455556666"
         ]
      },
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:queue1"
   }]
}
```

例4: ロールおよびユーザー名にクロスアカウントのアクセス許可を付与する

次のポリシー例ではrole1、米国東部 (オハイオ) リージョン123456789012/queue1で という名前のキューの共有アクセスを Amazon SQS が許可するすべてのアクションを使用するための111122223333クロスアカウントアクセス許可を username1 AWS アカウント および に付与します。

クロスアカウント権限は、次のアクションには適用されません。

- AddPermission
- CancelMessageMoveTask
- CreateQueue
- DeleteQueue
- ListMessageMoveTask
- ListQueues
- ListQueueTags
- RemovePermission
- SetQueueAttributes

- StartMessageMoveTask
- TagQueue
- UntagQueue

```
{
   "Version": "2012-10-17",
   "Id": "Queue1_Policy_UUID",
   "Statement": [{
      "Sid":"Queue1_AllActions",
      "Effect": "Allow",
      "Principal": {
         "AWS": [
            "arn:aws:iam::111122223333:role/role1",
            "arn:aws:iam::111122223333:user/username1"
         ]
      },
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:queue1"
   }]
}
```

例5:すべてのユーザーにアクセス権限を付与する

以下のサンプルポリシーは、すべてのユーザー (匿名ユーザー) に、111122223333/queue1 という名前のキューに対する ReceiveMessage アクセス権限を付与します。

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

例6:すべてのユーザーに時間制限付きのアクセス権限を付与する

以下のサンプルポリシーは、すべてのユーザー (匿名ユーザー) に、ReceiveMessageという名前のキューに対する111122223333/queue1アクセス権限を 2009 年 1 月 31 日の午後 12:00 (正午) から午後 3:00 の間のみ付与します。

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

例7:CIDR範囲のすべてのユーザーにすべてのアクセス権限を付与する

以下のサンプルポリシーは、すべてのユーザー (匿名ユーザー) に、 111122223333/queue1という名前のキューで共有できるすべてのAmazon SQS アクションを使用するアクセス権限を、リクエストが192.0.2.0/24 CIDRの範囲から生成された場合のみ付与します。

```
{
   "Version": "2012-10-17",
   "Id": "Queue1_Policy_UUID",
   "Statement": [{
      "Sid":"Queue1_AnonymousAccess_AllActions_AllowlistIP",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:*:111122223333:queue1",
      "Condition" : {
```

```
         "IpAddress" : {
            "aws:SourceIp":"192.0.2.0/24"
         }
      }
   }]
}
```

例8:異なる CIDR 範囲のユーザーの許可リストとブロックリストのアクセス権限

以下のサンプルポリシーには、2つのステートメントが含まれています。

- 最初のステートメントは、192.0.2.0/24CIDR の範囲 (192.0.2.188 を除く) に存在するすべてのユーザー (匿名ユーザー) に、SendMessage/queue1 という名前のキューに対する111122223333アクションを使用するアクセス権限を付与します。
- 2 番目のステートメントは、12.148.72.0/23CIDR の範囲に存在するすべてのユーザー (匿名ユーザー) のキュー使用をブロックます。

```
{
   "Version": "2012-10-17",
   "Id": "Queue1_Policy_UUID",
   "Statement": [{
      "Sid":"Queue1_AnonymousAccess_SendMessage_IPLimit",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:*:111122223333:queue1",
      "Condition" : {
         "IpAddress" : {
            "aws:SourceIp":"192.0.2.0/24"
         },
         "NotIpAddress" : {
            "aws:SourceIp":"192.0.2.188/32"
         }
      }
   }, {
      "Sid":"Queue1_AnonymousAccess_AllActions_IPLimit_Deny",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:*:111122223333:queue1",
      "Condition" : {
```

```
         "IpAddress" : {
            "aws:SourceIp":"12.148.72.0/23"
         }
      }
   }]
}
```

Amazon SQSアクセスポリシー言語を使用したカスタムポリシーの使用

AWS アカウント ID と基本的なアクセス許可 (SendMessageまたは の などReceiveMessage) のみに基づいて Amazon SQS アクセスを許可する場合は、独自のポリシーを作成する必要はありません。Amazon SQS AddPermission アクションを使用できます。

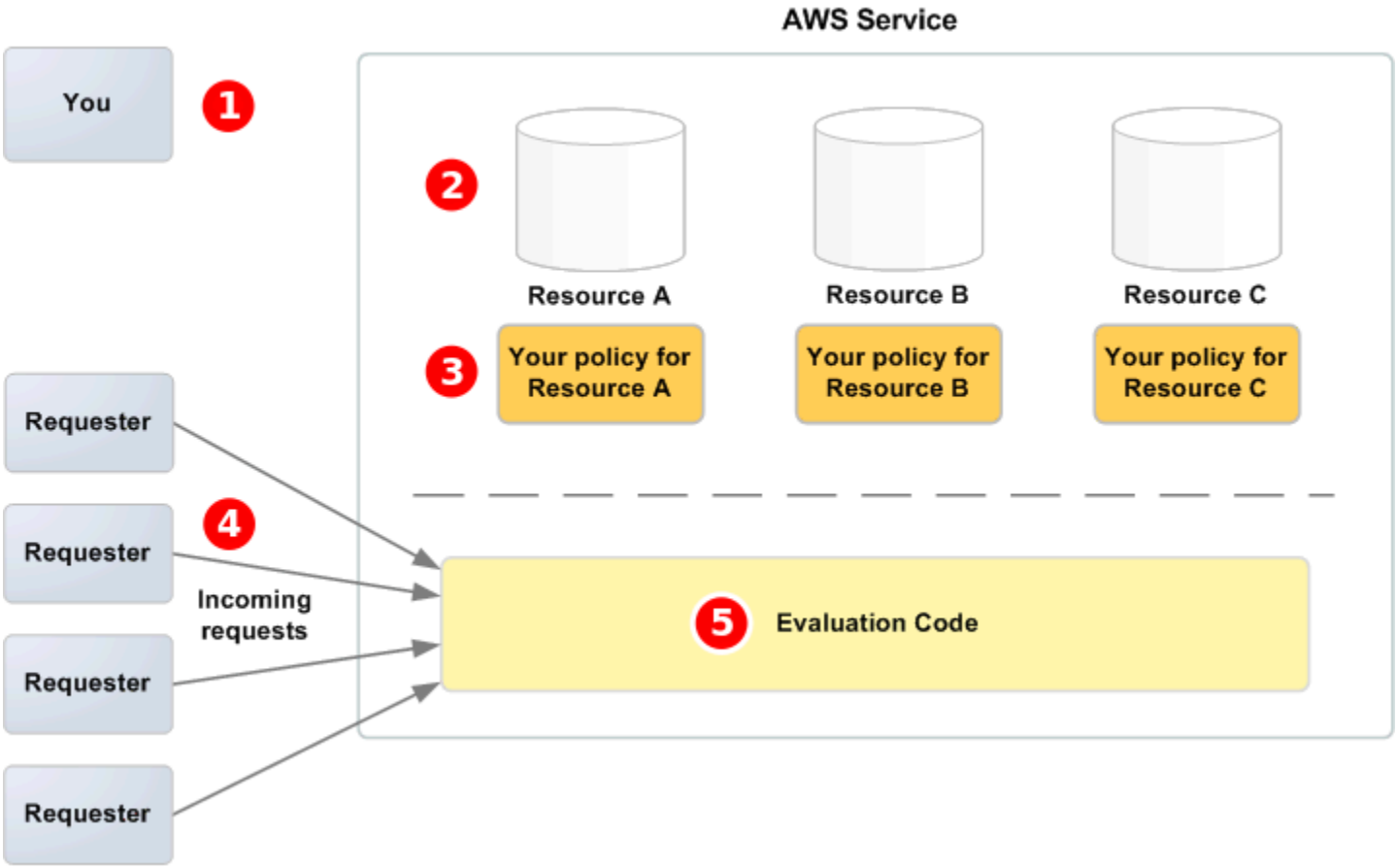
より具体的な条件 (リクエストが到着した時刻やリクエスタの IP アドレスなど) に基づいてアクセスを明示的に拒否または許可する場合は、独自の Amazon SQS ポリシーを記述し、Amazon SQS SetQueueAttributesアクションを使用して AWS システムにアップロードする必要があります。

トピック

- Amazon SQSのアクセスコントロールアーキテクチャ
- Amazon SQSアクセスコントロールプロセスのワークフロー
- Amazon SQSアクセスポリシー言語の主要な概念
- Amazon SQSアクセスポリシー言語評価ロジック
- Amazon SQSアクセスポリシー言語の明示的な拒否とデフォルトの拒否の関係
- Amazon SQS カスタムポリシーの制限事項
- カスタムの Amazon SQSアクセスポリシー言語の例

Amazon SQSのアクセスコントロールアーキテクチャ

以下の図は、Amazon SQSリソースのアクセスコントロールの説明です。



1 リソース所有者。

2 AWS サービスに含まれるリソース (Amazon SQS キューなど) 。

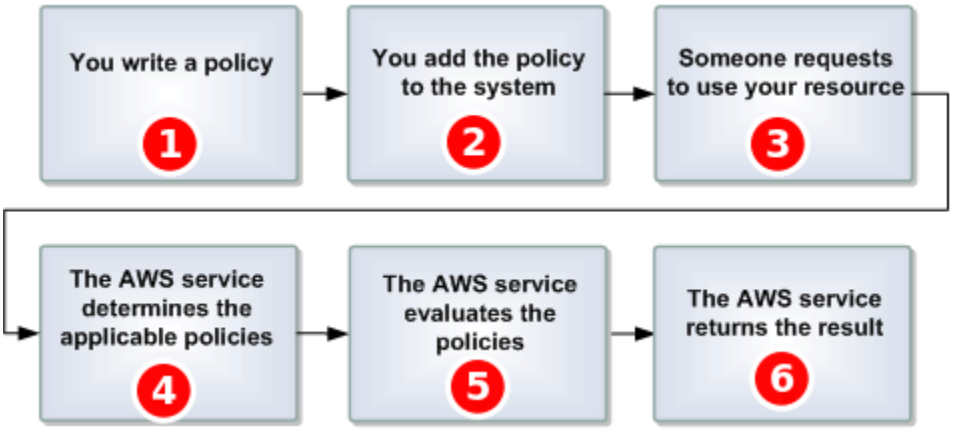
3 ポリシー。1 つのリソースごとに 1 つのポリシーを適用することをお勧めします。この AWS サービスは、ポリシーのアップロードと管理に使用する API を提供します。

4 リクエスタ、および AWS サービスに対する受信リクエスト。

5 アクセスポリシー言語の評価コード。これは、受信リクエストを該当するポリシーと照合して評価し、リクエスタがリソースにアクセスできるかどうかを判断する AWS サービス内のコードのセットです。

Amazon SQSアクセスコントロールプロセスのワークフロー

以下の図は、 Amazon SQSアクセスポリシー言語と アクセスコントロールの全般的なワークフローを表しています。



1 キューの Amazon SQSポリシーを記述します。

2 ポリシーを にアップロードします AWS。この AWS サービスは、ポリシーのアップロードに使用する API を提供します。たとえば、特定の Amazon SQSキュー用のポリシーをアップロードするために Amazon SQS`SetQueueAttributes`アクションを使用します。

3 ある人物から、Amazon SQSキューの使用を求めるリクエストが送信されます。

4 Amazon SQS はすべての利用可能な Amazon SQSポリシーを分析し、該当するものを判断します。

5 Amazon SQSがポリシーを評価し、キュー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。

6 ポリシーの評価結果に基づいて、Amazon SQSはリクエスタに `Access denied` エラーを返すか、リクエストの処理を続行します。

Amazon SQSアクセスポリシー言語の主要な概念

独自のポリシーを作成するには、JSON およびいくつかの重要な概念を理解している必要があります。

許可

Statementが[Effect] (効果)に設定されたallowの結果。

[アクション]

プリンシパルが実行アクセス権限を持っているアクティビティ。通常は AWSへのリクエスト。

Default-deny

Statementまたは許可設定を持たないExplicit-denyの結果。

条件

アクセス権限に関する制限または詳細。よく使用される条件は日時とIP アドレスに関連しています。

[Effect] (効果)

StatementのPolicyで評価時に返される結果。ポリシーステートメントを作成するときに、denyまたはallowの値を指定します。ポリシーの評価時に、Default-deny、許可、またはExplicit-denyの 3 つの結果が得られます。

Explicit-deny

Statementが[Effect] (効果)に設定されたdenyの結果。

評価

Amazon SQSが使用するプロセスでは、受信したリクエストを拒否または許可するかを、Policyに基づいて判断します。

Issuer

ユーザーはPolicyを記述して、リソースへのアクセス権限を付与します。定義上、発行者は常にリソース所有者です。Amazon AWS SQS ユーザーが所有していないリソースのポリシーを作成することを許可しません。 Amazon SQS

キー

アクセス制限に使用される基本項目です。

アクセス権限

条件およびキー を使用して、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。

Policy

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です。

Amazon SQSはポリシーを使用して、リソースに関してユーザーにアクセス権限を付与するかどうかを決定します。

プリンシパル

アクセス権限でPolicyを受け取るユーザー。

[リソース]

プリンシパル によるリクエストがアクセスするオブジェクト。

Statement

より広範なドキュメントの一部としてPolicyで作成された1つのアクセス権限の正式な説明。

リクエスタ

[リソース] にアクセスするためにリクエストを送信するユーザー。

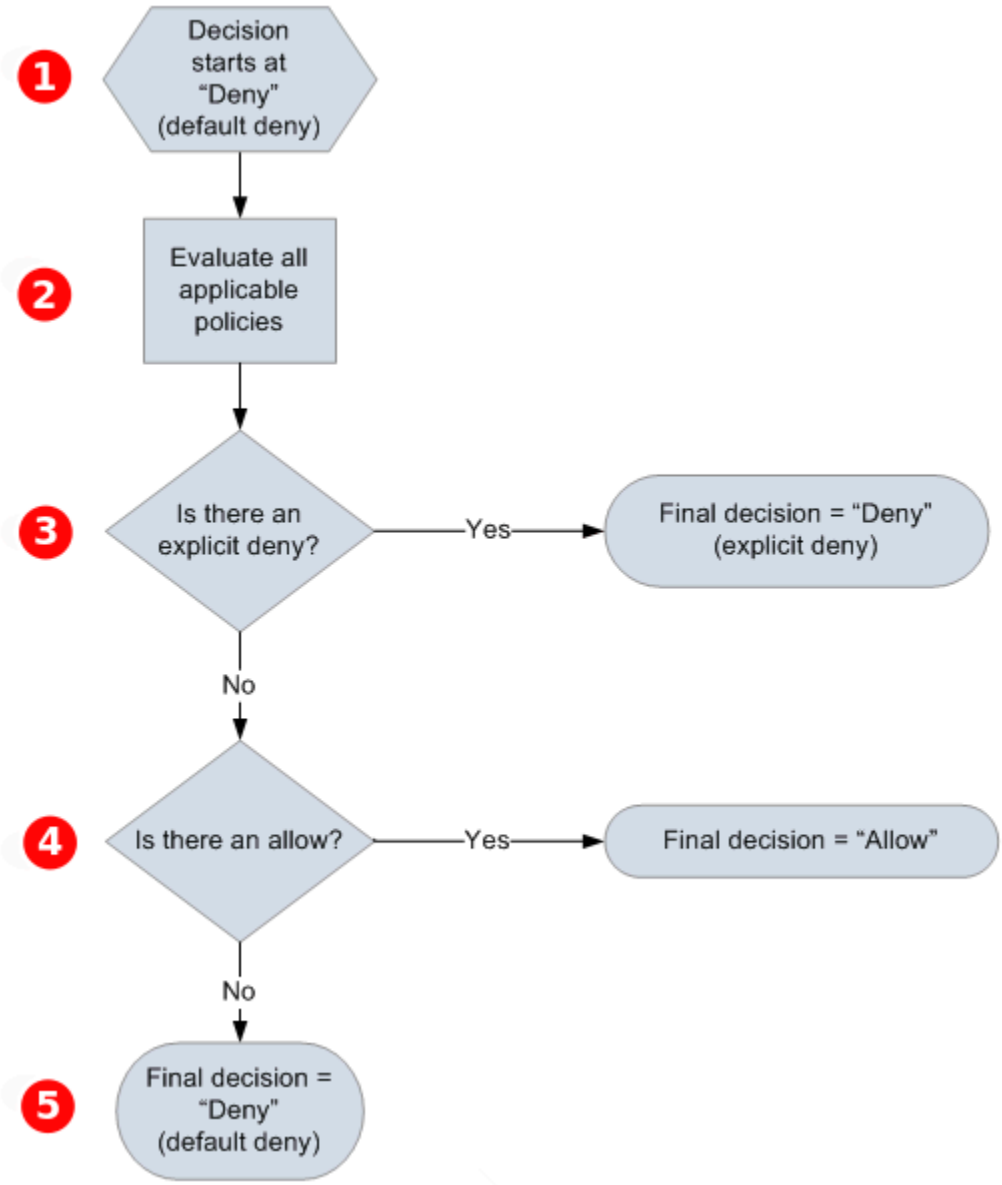
Amazon SQSアクセスポリシー言語評価ロジック

Amazon SQSは評価時に、リソース所有者以外の別の人物から与えられたリクエストに対し、許可するか拒否するかを決定します。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Default-denyをオーバーライドする。

- Explicit-deny はすべてのallowをオーバーライドする。
- ポリシー評価の順序は重要ではない。

次の図は、アクセス権限に関する決定事項をAmazon SQSがどのように評価するかに関する詳細を示しています。



1

決定はdefault-denyから始まります。

2

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

エンフォースメントコードは、リクエストに適用できる explicit-deny インストラクションを探します。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、プロセスは終了します。

4

explicit-deny が見つからなかった場合、リクエストに適応できるallow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)。

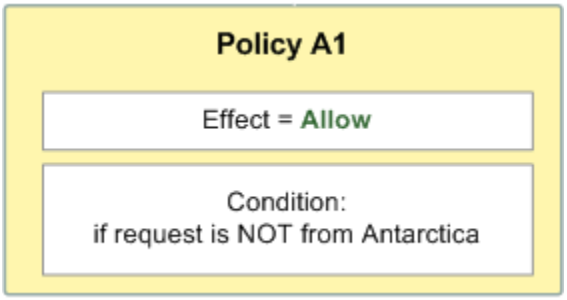
5

allowインストラクションが見つからなかった場合、最終決定は denyとなります(explicit-denyまたはallowが見つからない場合、default-denyとして見なされるためです)。

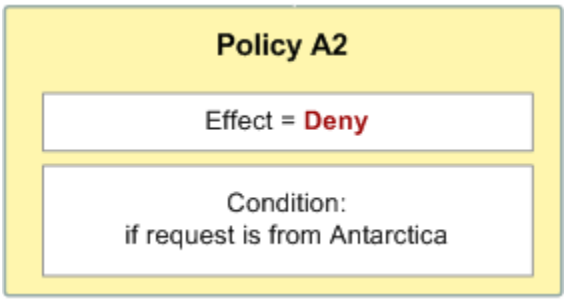
Amazon SQSアクセスポリシー言語の明示的な拒否とデフォルトの拒否の関係

Amazon SQS ポリシーがリクエストに直接適用されない場合、リクエストの結果は、Default-deny となります。たとえば、ユーザーが Amazon SQSを使用するアクセス権限をリクエストしたが、そのユーザーに適用される唯一のポリシーではDynamoDBを使用できる場合、リクエストの結果はdefault-denyとなります。

ステートメントの条件が満たされない場合、リクエストの結果は default-denyになります。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 [Effect] (効果) エレメントの値に基づいて、リクエストの結果は 許可 または Explicit-deny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 default-deny となります。たとえば、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 A1を記述します。以下の図はAmazon SQSポリシーについて解説しています。

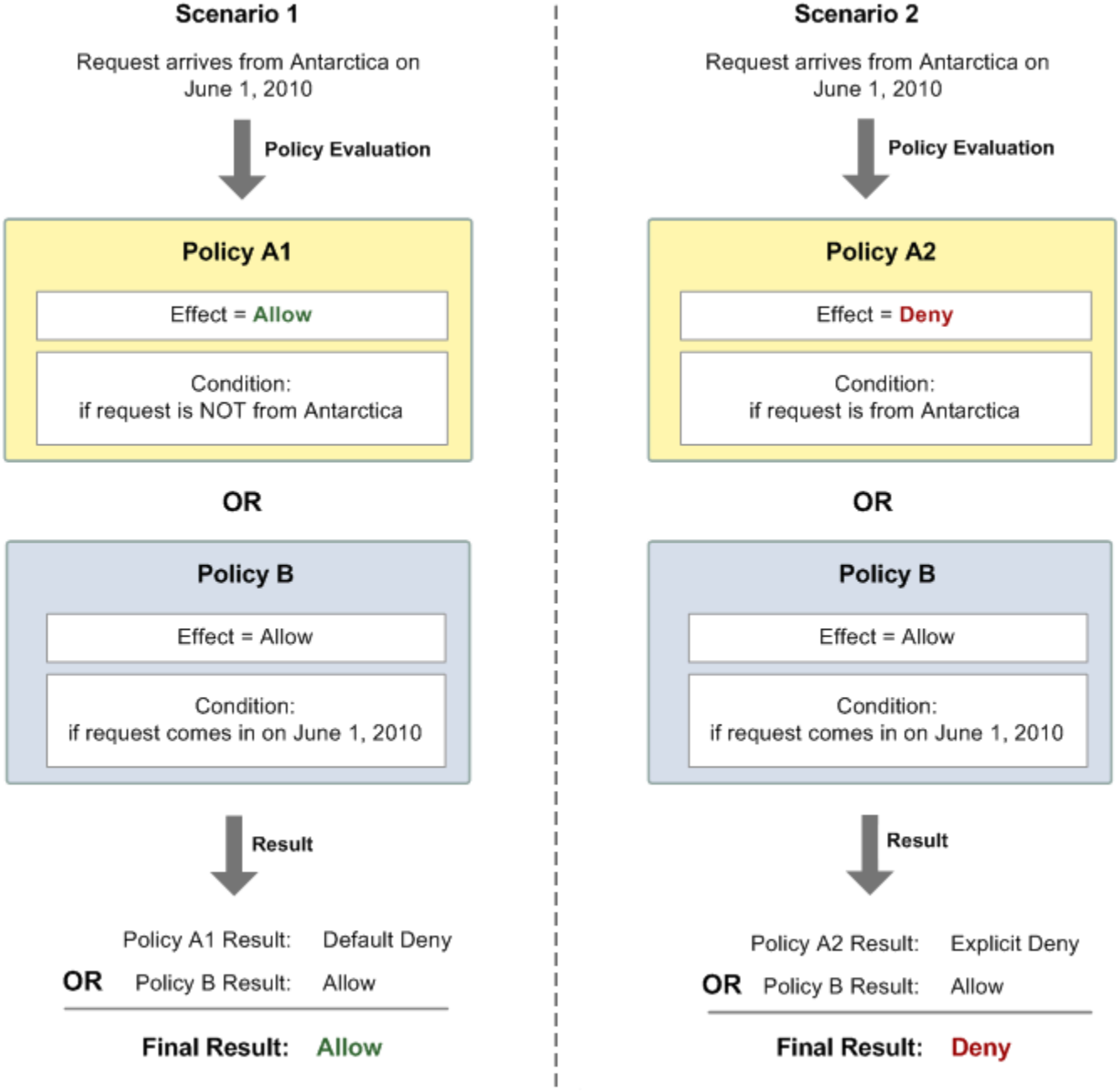


ユーザーがアメリカからリクエストを送信すると、条件が満たされ (リクエストは南極大陸から来ていない)、リクエストの結果は allow になります。ただし、ユーザーが南極からリクエストを送信した場合、条件は満たされず、リクエストはデフォルトで default-deny になります。南極大陸から来たリクエストを明示的に拒否するポリシー A2を作成して、結果を explicit-deny に変更することができます。以下の図はポリシーについて解説しています。



ユーザーが南極大陸からリクエストを送信すると、条件は満たされ、リクエストの結果はexplicit-denyとなります。

default-deny と explicit-deny の違いは重要です。allow は前者を上書きできますが、後者を上書きすることはできないためです。たとえば、ポリシー Bは、2010年6月1日に届いたリクエストを許可します。以下の図は、このポリシーをポリシー A1およびポリシー A2と組み合わせた場合を比較しています。



シナリオ 1では、ポリシー A1の結果はdefault-denyになり、ポリシー Bの結果はallowになります。これは、ポリシーで 2010年6月1日に来るリクエストが許可されるためです。ポリシー BによるallowはポリシーA1の default-deny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。

シナリオ 2 で、ポリシー B2の結果は explicit-deny になり、ポリシー Bの結果は allow になります。ポリシー A2 によるexplicit-denyは、ポリシー Bのallow 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 SQS カスタムポリシーの制限事項

クロスアカウントアクセス

クロスアカウント権限は、次のアクションには適用されません。

- AddPermission
- CancelMessageMoveTask
- CreateQueue
- DeleteQueue
- ListMessageMoveTask
- ListQueues
- ListQueueTags
- RemovePermission
- SetQueueAttributes
- StartMessageMoveTask
- TagQueue
- UntagQueue

条件キー

現在、Amazon SQS はIAMで使用可能な条件キーの制限されたサブセットのみをサポートしています。詳細については、「Amazon SQS (Amazon SQS)、 API のアクセス権限: アクションとリソースのリファレンスについて」を参照してください。

カスタムの Amazon SQSアクセスポリシー言語の例

一般的な Amazon SQSアクセスポリシーの例を次に示します。

例1:1つのアカウントにアクセス権限を与える

次のAmazon SQSポリシー例では、 AWS アカウント 444455556666で所有されるqueue2から送受信するアクセス権限を AWS アカウント 111122223333に与えます。

```
{
```

```
   "Version": "2012-10-17",
   "Id": "UseCase1",
   "Statement" : [{
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
         "AWS": [
            "111122223333"
         ]
      },
      "Action": [
         "sqs:SendMessage",
         "sqs:ReceiveMessage"
      ],
      "Resource": "arn:aws:sqs:us-east-2:444455556666:queue2"
   }]
}
```

例2:1つ以上のアカウントにアクセス権限を与える

次の Amazon SQS ポリシーの例では、特定の期間にアカウントが所有するキュー AWS アカウントへのアクセスを 1 つ以上許可します。このポリシーを作成し、Amazon SQSにアップロードするには、SetQueueAttributesのためアクションAddPermissionアクションでは、キューへのアクセスを許可するときに時間制限を指定することはできません。

```
{
   "Version": "2012-10-17",
   "Id": "UseCase2",
   "Statement" : [{
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
         "AWS": [
            "111122223333",
            "444455556666"
         ]
      },
      "Action": [
         "sqs:SendMessage",
         "sqs:ReceiveMessage"
      ],
      "Resource": "arn:aws:sqs:us-east-2:444455556666:queue2",
      "Condition": {
```

```
         "DateLessThan": {
            "AWS:CurrentTime": "2009-06-30T12:00Z"
         }
      }
   }]
}
```

例3:Amazon EC2インスタンスからのリクエストに許可を与える

次の例では、Amazon SQSポリシーはAmazon EC2 インスタンスから来るリクエストへのアクセスを許可します。この例では、"例2:1つ以上のアカウントにアクセス権限を与える" の例を作成します。2009年6月30日正午 (UTC) 以前のアクセスを制限し、IP 範囲203.0.113.0/24へのアクセスを制限します。このポリシーを作成し、Amazon SQS にアップロードするには、SetQueueAttributesのためにアクションAddPermissionアクションでは、キューへのアクセスを許可するときに IP アドレス制限を指定することはできません。

```
{
   "Version": "2012-10-17",
   "Id": "UseCase3",
   "Statement" : [{
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
         "AWS": [
            "111122223333"
         ]
      },
      "Action": [
         "sqs:SendMessage",
         "sqs:ReceiveMessage"
      ],
      "Resource": "arn:aws:sqs:us-east-2:444455556666:queue2",
      "Condition": {
         "DateLessThan": {
            "AWS:CurrentTime": "2009-06-30T12:00Z"
         },
         "IpAddress": {
            "AWS:SourceIp": "203.0.113.0/24"
         }
      }
   }]
}
```

例4:特定のアカウントへのアクセスを拒否する

次の Amazon SQS ポリシーの例では、キューへの特定の AWS アカウント アクセスを拒否します。この例では、例1:1つのアカウントにアクセス権限を与える「」の例に基づいて構築されています。指定された へのアクセスを拒否します AWS アカウント。このポリシーを作成し、Amazon SQS にアップロードするには、SetQueueAttributesのためにアクションAddPermissionアクションは、キューへのアクセスを拒否することを許可しません (キューへのアクセスのみを許可します) 。

```
{
   "Version": "2012-10-17",
   "Id": "UseCase4",
   "Statement" : [{
      "Sid": "1",
      "Effect": "Deny",
      "Principal": {
         "AWS": [
            "111122223333"
         ]
      },
      "Action": [
         "sqs:SendMessage",
         "sqs:ReceiveMessage"
      ],
      "Resource": "arn:aws:sqs:us-east-2:444455556666:queue2"
   }]
}
```

例5:VPC エンドポイントからではない場合はアクセスを拒否する

次のqueue1111122223333Amazon SQSポリシー例では、SendMessage へのアクセスを制限します。ReceiveMessage が vpce-1a2b3c4d および アクションを実行できるのは、VPC エンドポイント ID (aws:sourceVpce 条件を使用して指定) からのみです。詳細については、「Amazon SQS のAmazon Virtual Private Cloud エンドポイント」を参照してください。

> **Note**
>
> - aws:sourceVpce 条件では、VPC エンドポイントID のみが必要で、VPC エンドポイントリソースのARNは必要ありません。
> - 次の例を変更し、2番目のステートメントですべてのアクション (sqs:*) を拒否して、特定の VPC エンドポイントにすべての Amazon SQS アクションを制限できます。ただし、

このようなポリシーステートメントにより、すべてのアクション (キューのアクセス権限を変更するために必要な管理アクションを含む) が、ポリシーで定義された特定の VPC エンドポイントを通じて行われる必要があることが規定されます。この場合、ユーザーは今後キューのアクセス権限を変更できなくなる可能性があります。

```
{
   "Version": "2012-10-17",
   "Id": "UseCase5",
   "Statement": [{
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
         "AWS": [
            "111122223333"
         ]
      },
      "Action": [
         "sqs:SendMessage",
         "sqs:ReceiveMessage"
      ],
         "Resource": "arn:aws:sqs:us-east-2:111122223333:queue1"
      },
      {
         "Sid": "2",
         "Effect": "Deny",
         "Principal": "*",
         "Action": [
            "sqs:SendMessage",
            "sqs:ReceiveMessage"
         ],
         "Resource": "arn:aws:sqs:us-east-2:111122223333:queue1",
         "Condition": {
            "StringNotEquals": {
               "aws:sourceVpce": "vpce-1a2b3c4d"
            }
         }
      }
   ]
}
```

Amazon SQSで一時的なセキュリティ認証情報を使用する

IAM では、独自のセキュリティ認証情報を使用してユーザーを作成するだけでなく、任意のユーザーに一時的なセキュリティ認証情報を付与して、ユーザーが AWS サービスとリソースにアクセスできるようにします。 AWS アカウントを持つユーザーを管理できます。また、 を持たないシステムのユーザー AWS アカウント (フェデレーティッドユーザー) を管理することもできます。さらに、 AWS リソースにアクセスするために作成したアプリケーションは、「ユーザー」と見なされることもあります。

Amazon SQSに対するリクエストを作成するときに、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 SQS はリクエストを拒否します。

> **Note**
>
> 一時的な認証情報に基づいてポリシーを設定することはできません。

前提条件

1. 一時的なセキュリティ認証情報を作成するには、IAMを使用します。
   - セキュリティトークン
   - アクセスキー ID
   - シークレットアクセスキー
2. 一時アクセスキー IDとセキュリティトークンで署名対象のリクエスト文字列を準備します。
3. 独自のシークレットアクセスキーの代わりに一時シークレットアクセスキーを使用して、クエリ API リクエストに署名します。

> **Note**
>
> 署名付きのクエリAPI リクエストを送信するときは、独自のアクセスキー ID の代わりに一時アクセスキー IDを使用して、セキュリティトークンを含めます。一時的なセキュリティ認証情報の IAM サポートの詳細については、IAM ユーザーガイドの「 AWS リソースへの一時的なアクセスの付与」を参照してください。

一時的なセキュリティ認証情報を使用して Amazon SQSクエリ API アクションを呼び出すには

1. を使用して一時的なセキュリティトークンをリクエストします AWS Identity and Access Management。詳細については、IAM ユーザーガイドの「一時的なセキュリティ認証情報を使用して、IAMユーザーのアクセスを可能にする」 を参照してください。

   IAMにより、セキュリティトークン、アクセスキー ID、シークレットアクセスキーが返信されます。

2. クエリは、独自のアクセスキー IDの代わりに一時アクセスキー IDを使用し、セキュリティトークンを含めて準備します。独自のシークレットアクセスキーの代わりに一時シークレットアクセスキーを使用してリクエストに署名します。

3. 一時アクセスキー IDとセキュリティトークンを含む署名付きクエリ文字列を送信します。

   以下の例は、一時的なセキュリティ認証情報を使用してAmazon SQS リクエストを認証する方法を示しています。*AUTHPARAMS*の構造はAPIリクエストの署名によって異なります。詳細については、Amazon Web Services 全般のリファレンスのAWS 「 API リクエストの署名」を参照してください。

```
https://sqs.us-east-2.amazonaws.com/
?Action=CreateQueue
&DefaultVisibilityTimeout=40
&QueueName=MyQueue
&Attribute.1.Name=VisibilityTimeout
&Attribute.1.Value=40
&Expires=2020-12-18T22%3A52%3A43PST
&SecurityToken=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Version=2012-11-05
&AUTHPARAMS
```

   以下の例は、一時的なセキュリティ認証情報を使用し、SendMessageBatch アクションで 2 つのメッセージを送信する方法を示しています。

```
https://sqs.us-east-2.amazonaws.com/
?Action=SendMessageBatch
&SendMessageBatchRequestEntry.1.Id=test_msg_001
&SendMessageBatchRequestEntry.1.MessageBody=test%20message%20body%201
&SendMessageBatchRequestEntry.2.Id=test_msg_002
&SendMessageBatchRequestEntry.2.MessageBody=test%20message%20body%202
&SendMessageBatchRequestEntry.2.DelaySeconds=60
```

```
&Expires=2020-12-18T22%3A52%3A43PST
&SecurityToken=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
&AWSAccessKeyId=AKIAI44QH8DHBEXAMPLE
&Version=2012-11-05
&AUTHPARAMS
```

最小特権ポリシーを持つ暗号化された Amazon SQS キューのアクセス管理

Amazon SQS では、AWS Key Management Service (KMS) と統合されたサーバー側の暗号化 (SSE) を使用して、アプリケーション間で機密データを交換できます。Amazon SQS と の統合により AWS KMS、Amazon SQS を保護するキーと、他の AWS リソースを保護するキーを一元管理できます。

複数の AWS サービスがAmazon SQSにイベントを送信するイベントソースとして機能します。イベントソースが暗号化された Amazon SQS キューにアクセスできるようにするには、カスタマーマネージド AWS KMS キーを使用してキューを設定する必要があります。次に、キーポリシーを使用して、サービスが必要な AWS KMS API メソッドを使用できるようにします。サービスには、キューがイベントを送信できるようにするためのアクセス認証のアクセス権限も必要です。これを実現するには、Amazon SQS ポリシーを使用します。Amazon SQS ポリシーは、Amazon SQS キューとそのデータへのアクセスを制御するために使用できるリソースベースのポリシーです。

以下のセクションでは、Amazon SQS ポリシーと AWS KMS キーポリシーを使用して、暗号化された Amazon SQS キューへのアクセスを制御する方法について説明します。このガイドのポリシーは、最小特権を達成するのに役立ちます。

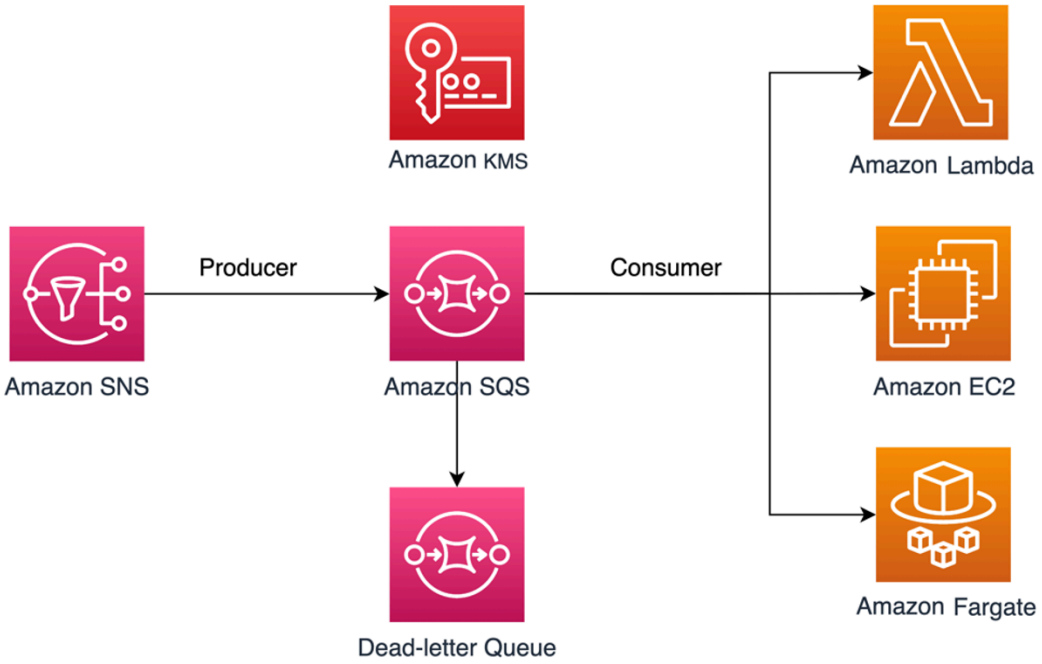
また、このガイドでは、`aws:SourceArn`、`aws:SourceAccount`、および `aws:PrincipalOrgID` グローバル IAM 条件コンテキストキーを使用して、リソースベースのポリシーで混乱した使節の問題に対処する方法についても説明します。

トピック

- 概要
- Amazon SQS の最小特権キーポリシー
- デッドレターキューの Amazon SQS ポリシーステートメント
- サービス間での混乱した使節問題を防止する
- IAM Access Analyzer を使用して、クロスアカウントアクセスを確認します。

概要

このトピックでは、一般的なユースケースを順を追って説明し、キーポリシーと Amazon SQS キューポリシーを構築する方法について説明します。このユースケースは次の画像に示されています。



この例では、メッセージプロデューサーは Amazon Simple Notification Service (SNS) トピックで、暗号化された Amazon SQS キューにメッセージをファンアウトするように設定されています。メッセージコンシューマーは、AWS Lambda 関数、Amazon Elastic Compute Cloud (EC2) インスタンス、AWS Fargate コンテナなどのコンピューティングサービスです。Amazon SQS キューは、失敗したメッセージをデッドレターキュー (DLQ) に送信するように設定されます。DLQ は、未使用のメッセージを分離して、処理が成功しない理由を調べることができるため、アプリケーションやメッセージングシステムのデバッグに役立ちます。このトピックで定義されているソリューションでは、Lambda 関数などのコンピューティングサービスを使用して Amazon SQS キューに保存されたメッセージを処理します。メッセージコンシューマーが仮想プライベートクラウド (VPC) に配置されている場合、このガイドに含まれる `DenyReceivingIfNotThroughVPCE` ポリシーステートメントにより、メッセージの受信をその特定の VPC に制限できます。

> **Note**
>
> このガイドには、必要な IAM アクセス許可のみがポリシーステートメントの形式で記載されています。ポリシーを作成するには、Amazon SQS ポリシーまたは AWS KMS キーポリシーにステートメントを追加する必要があります。このガイドでは、Amazon SQS キュー

または AWS KMS キーの作成方法については説明していません。これらのリソースの作成方法については、「Amazon SQS キューを作成する」および「キーの作成」を参照してください。
このガイドで定義されている Amazon SQS ポリシーでは、メッセージを同じ Amazon SQS キューまたは別の Amazon SQS キューに直接リドライブすることはサポートされません。

Amazon SQS の最小特権キーポリシー

このセクションでは、Amazon SQS キューの暗号化に使用するカスタマーマネージドキー AWS KMS の で必要な最小特権のアクセス許可について説明します。これらのアクセス許可により、最小特権を実装しながら、アクセスを目的のエンティティのみに制限できます。キーポリシーは、以下のポリシーステートメントで構成されている必要があります。詳細については以下で説明します。

- AWS KMS キーに管理者権限を付与する
- キーメタデータへの読み取り専用アクセスを付与する
- キューにメッセージを発行するために、Amazon SNS KMS のアクセス許可を Amazon SNS に付与する
- コンシューマーがキューからのメッセージを復号化することを許可する

AWS KMS キーに管理者権限を付与する

AWS KMS キーを作成するには、 AWS KMS キーのデプロイに使用する IAM ロールに AWS KMS 管理者権限を付与する必要があります。これらの管理者権限は、以下の `AllowKeyAdminPermissions` ポリシーステートメントで定義されています。このステートメントを AWS KMS キーポリシーに追加するときは、*`<admin-role ARN>`* を AWS KMS 、キーのデプロイ、 AWS KMS キーの管理、またはその両方に使用される IAM ロールの Amazon リソースネーム (ARN) に置き換えてください。これは、デプロイパイプラインの IAM ロールでも、AWS Organizations 内の組織の管理者ロールでもかまいません。

```
{
  "Sid": "AllowKeyAdminPermissions",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "<admin-role ARN>"
    ]
  },
```

```
  "Action": [
    "kms:Create*",
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:TagResource",
    "kms:UntagResource",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

> **Note**
>
> AWS KMS キーポリシーでは、 Resource要素の値は である必要があります。つまり*、「この AWS KMS キー」です。アスタリスク (*) は、 AWS KMS キーポリシーがアタッチされているキーを識別します。

キーメタデータへの読み取り専用アクセスを付与する

他の IAM ロールにキーメタデータへの読み取り専用アクセス権を付与するには、その AllowReadAccessToKeyMetaData ステートメントをキーポリシーに追加します。例えば、次のステートメントでは、監査目的でアカウント内のすべての AWS KMS キーを一覧表示できます。このステートメントは、キーメタデータへの読み取り専用アクセスを AWS ルートユーザーに付与します。したがって、アカウント内の IAM プリンシパルは、その ID ベースのポリシーに次のステートメント (kms:Describe*、kms:Get*、kms:List*) に記載されているアクセス権限が付与されていれば、キーメタデータにアクセスできます。*<account-ID>* をユーザー自身の情報に必ず置き換えます。

```
{
  "Sid": "AllowReadAcesssToKeyMetaData",
  "Effect": "Allow",
  "Principal": {
```

```
    "AWS": [
      "arn:aws:iam::<accountID>:root"
    ]
  },
  "Action": [
    "kms:Describe*",
    "kms:Get*",
    "kms:List*"
  ],
  "Resource": "*"
}
```

キューにメッセージを発行するために、Amazon SNS KMS のアクセス許可を Amazon SNS に付与する

Amazon SNS トピックが、暗号化された Amazon SQS キューにメッセージを発行できるようにするには、AllowSNSToSendToSQS ポリシーステートメントをキーポリシーに追加します。このステートメントは、 AWS KMS キーを使用して Amazon SNSAmazon SQSに付与します。*<account-ID>* をユーザー自身の情報に必ず置き換えます。

> **Note**
>
> ステートメントConditionの は、同じ AWS アカウントの Amazon SNS サービスのみへのアクセスを制限します。

```
{
  "Sid": "AllowSNSToSendToSQS",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "sns.amazonaws.com"
    ]
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
```

```
      "aws:SourceAccount": "<account-id>"
    }
  }
}
```

コンシューマーがキューからのメッセージを復号化することを許可する

次の AllowConsumersToReceiveFromTheQueue ステートメントは、暗号化された Amazon SQS キューから受信したメッセージを復号化するために必要なアクセス許可を Amazon SQS メッセージコンシューマーに付与します。ポリシーステートメントを添付するときは、*<consumer's runtime role ARN>* をメッセージコンシューマーの IAM ランタイムロール ARN に置き換えます。

```
{
  "Sid": "AllowConsumersToReceiveFromTheQueue",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "<consumer's execution role ARN>"
    ]
  },
  "Action": [
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

Amazon SQS の最小特権ポリシー

このセクションでは、このガイドの対象となるユースケース (例えば Amazon SNS から Amazon SQS へ) の最小特権の Amazon SQS キューポリシーについて説明します。定義されたポリシーは、Deny および Allow ステートメントの両方を組み合わせて使用することで、意図しないアクセスを防ぐように設計されています。Allow ステートメントは、目的の 1 つ以上のエンティティへのアクセスを許可します。Deny ステートメントは、ポリシー条件の範囲内で目的のエンティティを除外しながら、意図しない他のエンティティが Amazon SQS キューにアクセスするのを防ぎます。

Amazon SQS ポリシーには以下のステートメントが含まれています。詳細については以下で説明します。

- Amazon SQS の管理権限を制限する
- 指定された組織からの Amazon SQS キューアクションを制限する

- Amazon SQS のアクセス許可をコンシューマーに付与する
- 転送時の暗号化を強制する
- メッセージ送信を特定の Amazon SNS トピックに制限する
- (オプション) 特定の VPC エンドポイントに対するメッセージの受信を制限する

Amazon SQS の管理権限を制限する

次の RestrictAdminQueueActions ポリシーステートメントは、Amazon SQS の管理権限を、キューのデプロイ、キューの管理、またはその両方に使用する 1 つまたは複数の IAM ロールのみに制限します。*<placeholder values>* をユーザー自身の情報に必ず置き換えます。Amazon SQS キューのデプロイに使用する IAM ロールの ARN と、Amazon SQS 管理権限が必要なすべての管理者ロールの ARN を指定します。

```
{
  "Sid": "RestrictAdminQueueActions",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:AddPermission",
    "sqs:DeleteQueue",
    "sqs:RemovePermission",
    "sqs:SetQueueAttributes"
  ],
  "Resource": "<SQS Queue ARN>",
  "Condition": {
    "StringNotLike": {
      "aws:PrincipalARN": [
        "arn:aws:iam::<account-id>:role/<deployment-role-name>",
        "<admin-role ARN>"
      ]
    }
  }
}
```

指定された組織からの Amazon SQS キューアクションを制限する

Amazon SQS リソースを外部アクセス (AWS 組織外のエンティティによるアクセス) から保護するには、次のステートメントを使用します。このステートメントは、Amazon SQS キューアクセスを

Condition で指定した組織に制限します。必ず *<SQS queue ARN>* Amazon SQS キューのデプロイに使用した IAM ロールの ARN に、*<org-id>* を組織 ID に置き換えてください。

```
{
  "Sid": "DenyQueueActionsOutsideOrg",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:AddPermission",
    "sqs:ChangeMessageVisibility",
    "sqs:DeleteQueue",
    "sqs:RemovePermission",
    "sqs:SetQueueAttributes",
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "StringNotEquals": {
      "aws:PrincipalOrgID": [
        "<org-id>"
      ]
    }
  }
}
```

Amazon SQS のアクセス許可をコンシューマーに付与する

Amazon SQS キューからメッセージを受信するには、メッセージコンシューマーに必要なアクセス許可を与える必要があります。次のポリシーステートメントは、Amazon SQS キューからのメッセージを消費するために必要なアクセス許可を、指定したコンシューマーに付与します。Amazon SQS ポリシーにステートメントを追加する際は、必ず *<consumer's IAM runtime role ARN>* をコンシューマーが使用する IAM ランタイムロールの ARN に置き換え、*<SQS queue ARN>* を Amazon SQS キューをデプロイするために使用される IAM ロールの ARN に置き換えてください。

```
{
  "Sid": "AllowConsumersToReceiveFromTheQueue",
  "Effect": "Allow",
  "Principal": {
    "AWS": "<consumer's IAM execution role ARN>"
  },
```

```
  "Action": [
    "sqs:ChangeMessageVisibility",
    "sqs:DeleteMessage",
    "sqs:GetQueueAttributes",
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>"
}
```

他のエンティティが Amazon SQS キューからメッセージを受信しないようにするには、Amazon SQS キューポリシーに DenyOtherConsumersFromReceiving ステートメントを追加します。このステートメントは、メッセージの消費を指定したコンシューマーに制限します。つまり、そのコンシューマーの ID アクセス許可によってアクセスが許可される場合でも、他のコンシューマーにはアクセスできなくなります。*<SQS queue ARN>* と *<consumer's runtime role ARN>* をユーザー自身の情報に必ず置き換えてください。

```
{
  "Sid": "DenyOtherConsumersFromReceiving",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:ChangeMessageVisibility",
    "sqs:DeleteMessage",
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "StringNotLike": {
      "aws:PrincipalARN": "<consumer's execution role ARN>"
    }
  }
}
```

転送時の暗号化を強制する

以下の DenyUnsecureTransport ポリシーステートメントは、コンシューマーとプロデューサーが安全なチャネル (TLS 接続) を使用して Amazon SQS キューからメッセージを送受信することを強

制します。*<SQS queue ARN>* を Amazon SQS キューのデプロイに使用した IAM ロールの ARN に必ず置き換えてください。

```
{
  "Sid": "DenyUnsecureTransport",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": [
    "sqs:ReceiveMessage",
    "sqs:SendMessage"
  ],
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "Bool": {
      "aws:SecureTransport": "false"
    }
  }
}
```

メッセージ送信を特定の Amazon SNS トピックに制限する

以下の AllowSNSToSendToTheQueue ポリシーステートメントは、Amazon SNS トピックが指定された Amazon SQS キューへメッセージを送信できるようにします。*<SQS queue ARN>* を Amazon SQS キューのデプロイに使用した IAM ロールの ARN に、*<SNS topic ARN>* を Amazon SNS トピック ARN に必ず置き換えてください。

```
{
  "Sid": "AllowSNSToSendToTheQueue",
  "Effect": "Allow",
  "Principal": {
    "Service": "sns.amazonaws.com"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "ArnLike": {
      "aws:SourceArn": "<SNS topic ARN>"
    }
```

```
  }
}
```

次の DenyAllProducersExceptSNSFromSending ポリシーステートメントは、他のプロデューサーがキューにメッセージを送信できないようにします。*<SQS queue ARN>* と*<SNS topic ARN>* をユーザー自身の情報に置き換えてください。

```
{
  "Sid": "DenyAllProducersExceptSNSFromSending",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS queue ARN>",
  "Condition": {
    "ArnNotLike": {
      "aws:SourceArn": "<SNS topic ARN>"
    }
  }
}
```

(オプション) 特定の VPC エンドポイントに対するメッセージの受信を制限する

メッセージの受信を特定の VPC エンドポイントのみに制限するには、以下のポリシーステートメントを Amazon SQS キューポリシーに追加します。このステートメントは、メッセージが目的の VPC エンドポイントからのものでない限り、メッセージコンシューマーがキューからメッセージを受信することを防ぎます。*<SQS queue ARN>* を Amazon SQS キューのデプロイに使用した IAM ロールの ARN に、および*<vpce_id>* を VPC エンドポイントの ID に置き換えてください。

```
{
  "Sid": "DenyReceivingIfNotThroughVPCE",
  "Effect": "Deny",
  "Principal": "*",
  "Action": [
    "sqs:ReceiveMessage"
  ],
  "Resource": "<SQS queue ARN>",
```

```
  "Condition": {
    "StringNotEquals": {
      "aws:sourceVpce": "<vpce id>"
    }
  }
}
```

デッドレターキューの Amazon SQS ポリシーステートメント

ステートメント ID で識別される以下のポリシーステートメントを DLQ アクセスポリシーに追加します。

- RestrictAdminQueueActions
- DenyQueueActionsOutsideOrg
- AllowConsumersToReceiveFromTheQueue
- DenyOtherConsumersFromReceiving
- DenyUnsecureTransport

前述のポリシーステートメントを DLQ アクセスポリシーに追加することに加えて、次のセクションで説明するように、Amazon SQS キューへのメッセージ送信を制限するステートメントも追加する必要があります。

Amazon SQS キューへのメッセージの送信を制限する

同じアカウントの Amazon SQS キューのみへのアクセスを制限するには、DLQ キューポリシーに次の DenyAnyProducersExceptSQS ポリシーステートメントを追加します。DLQ はメインキューを作成する前にデプロイする必要があるため、このステートメントでは特定のキューへのメッセージ送信を制限しません。DLQ の作成時に Amazon SQS ARN を知ることができないためです。1 つの Amazon SQS キューのみにアクセスを制限する必要がある場合、Condition 内の aws:SourceArn を Amazon SQS ソースキューの ARN で変更します。

```
{
  "Sid": "DenyAnyProducersExceptSQS",
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Action": "sqs:SendMessage",
  "Resource": "<SQS DLQ ARN>",
```

```
  "Condition": {
    "ArnNotLike": {
      "aws:SourceArn": "arn:aws:sqs:<region>:<account-id>:*"
    }
  }
}
```

> **Important**
>
> このガイドで定義されている Amazon SQS キューポリシーは、sqs:PurgeQueue アクションを特定の IAM ロールに制限しません。sqs:PurgeQueue アクションにより、Amazon SQS キューからのすべてのメッセージの削除が可能になります。このアクションを使用して、Amazon SQS キューを置き換えずにメッセージ形式を変更することもできます。アプリケーションをデバッグするときに、Amazon SQS キューをクリアして、エラーの可能性のあるメッセージを削除できます。アプリケーションをテストするときは、Amazon SQS キューに大量のメッセージを送り、本番環境に入る前にキューを消去して最初からやり直すことができます。このアクションを特定のロールに制限しない理由は、Amazon SQS キューをデプロイする際にこのロールがわからない可能性があるためです。キューを削除するには、このアクセス許可をロールの ID ベースのポリシーに追加する必要があります。

サービス間での混乱した使節問題を防止する

「混乱した代理」問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。これを防ぐために、 は、アカウント内のリソースへのアクセスをサードパーティー (クロスアカウントと呼ばれる) または他の AWS サービス (クロスサービスと呼ばれる) に提供する場合、アカウントを保護するのに役立つツール AWS を提供します。このセクションのポリシーステートメントは、サービス間の混乱した使節の問題を防止するのに役立ちます。

サービス間でのなりすましは、あるサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスが操作され、それ自体のアクセス許可を通じて、別の顧客のリソースに対して本来アクセス許可が付与されるべきではない形で働きかけが行われることがあります。この問題を防ぐため、この記事で定義されているリソースベースのポリシーでは、aws:SourceArn、aws:SourceAccount、aws:PrincipalOrgID およびグローバル IAM 条件コンテキストキーを使用しています。これにより、サービスが持つアクセス許可が、 AWS Organizations 内の特定のリソース、特定のアカウント、または特定の組織に制限されます。

IAM Access Analyzer を使用して、クロスアカウントアクセスを確認します。

AWS IAM Access Analyzer を使用してAmazon SQSキューポリシーと AWS KMS キーポリシーを確認し、Amazon SQS キューまたは AWS KMS キーが外部エンティティにアクセスを許可すると警告できます。IAM Access Analyzer の機能は、信頼ゾーン外のエンティティと共有されている組織とアカウントのリソースを識別するのに役立ちます。この信頼ゾーンは、IAM Access Analyzer を有効にするときに指定する AWS アカウントまたは AWS Organizations 内の組織です。

IAM Access Analyzer は、ロジックベースの推論を使用して AWS 環境内のリソースベースのポリシーを分析することで、外部プリンシパルと共有されているリソースを識別します。信頼ゾーン外で共有されているリソースのインスタンスごとに、Access Analyzer は結果を生成します。結果には、アクセスと付与される外部プリンシパルに関する情報が含まれます。結果を確認して、アクセスが意図的で安全なものであるか、またはアクセスが意図しないものであるか、セキュリティ上のリスクであるかを判断します。意図しないアクセスについては、影響を受けるポリシーを確認して修正します。 AWS IAM Access Analyzer が AWS リソースへの意図しないアクセスを識別する方法の詳細については、このブログ記事を参照してください。

AWS IAM Access Analyzer の詳細については、AWS 「IAM Access Analyzer ドキュメント」を参照してください。

Amazon SQS (Amazon SQS)、 API のアクセス権限: アクションとリソースのリファレンスについて

アクセスコントロール をセットアップし、IAMアイデンティティにアタッチできるアクセス権限ポリシーを作成するときは、以下の表をリファレンスとして使用できます。には、各 Amazon Simple Queue Service アクション、アクションを実行するためのアクセス許可を付与できる対応するアクション、およびアクセス許可を付与できる AWS リソースが含まれます。

ポリシーのActionフィールドでアクションを指定し、ポリシーのResourceフィールドでリソースの値を指定します。アクションを指定するには、sqs:プレフィックスに続けて アクション名を使用します (例:sqs:CreateQueue)。

現在、Amazon SQS は IAM で使用可能なグローバル条件コンテキストキーをサポートしています。

Amazon Simple キューサービス API およびアクション用のアクセス権限が必要です。

AddPermission

アクション:sqs:AddPermission

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

ChangeMessage可視性

アクション:sqs:ChangeMessageVisibility

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

ChangeMessageVisibilityBatch

アクション:sqs:ChangeMessageVisibilityBatch

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

CreateQueue

アクション:sqs:CreateQueue

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

DeleteMessage

アクション:sqs:DeleteMessage

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

DeleteMessageバッチ

アクション:sqs:DeleteMessageBatch

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

DeleteQueue

アクション:sqs:DeleteQueue

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

GetQueue属性

アクション:sqs:GetQueueAttributes

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

GetQueueURL

アクション:sqs:GetQueueUrl

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

ListDeadLetterSourceキュー

アクション:sqs:ListDeadLetterSourceQueues

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

ListQueues

アクション:sqs:ListQueues

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

ListQueueタグ

アクション:sqs:ListQueueTags

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

PurgeQueue

アクション:sqs:PurgeQueue

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

ReceiveMessage

アクション:sqs:ReceiveMessage

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

RemovePermission

アクション:sqs:RemovePermission

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

SendMessage および SendMessageバッチ

アクション:sqs:SendMessage

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

SetQueue属性

アクション:sqs:SetQueueAttributes

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

TagQueue

アクション:sqs:TagQueue

リソース: arn:aws:sqs:*region*:*account_id*:*queue_name*

UntagQueue

アクション:sqs:UntagQueue

リソース:arn:aws:sqs:*region*:*account_id*:*queue_name*について

Amazon SQSのロギングとモニタリング

このセクションでは、 を使用して API コール CloudTrail をキャプチャする方法や、キューの Amazon SQS のログ記録とモニタリングオプションについて説明します。 CloudWatch

トピック

- AWS CloudTrailを使用したAmazon SQS (Amazon SQS)(API)コールのロギング
- を使用した Amazon SQS キューのモニタリング CloudWatch

AWS CloudTrailを使用したAmazon SQS (Amazon SQS)(API)コールのロギング

Amazon SQS は と統合 AWS CloudTrail され、ユーザー、ロール、または AWS service. CloudTrail captures API コール Amazon SQS をイベント として記録します。これには、Amazon SQS コンソールを介して開始されたインタラクションやAmazon SQS Amazon SQS APIs の呼び出しを介してプログラム的に開始されたインタラクションが含まれます。

トピック

- の Amazon SQS 情報 CloudTrail
- の管理イベント CloudTrail
- のデータイベント CloudTrail
- 例: Amazon SQS CloudTrail の管理イベント
- 例:Amazon SQS CloudTrail のデータイベント

の Amazon SQS 情報 CloudTrail

CloudTrail AWS アカウントを作成すると、 はデフォルトでオンになっています。サポートされている Amazon SQS イベントアクティビティが発生すると、 CloudTrail イベント履歴の他の AWS サー

ビスイベントとともにイベントに記録されます。 AWS アカウントの最近のイベントを表示、検索、ダウンロードできます。詳細については、「 AWS CloudTrail ユーザーガイド」の「イベント履歴を含む CloudTrail イベントの表示」を参照してください。

などのキュー管理オペレーションを呼び出す Amazon SQS APIs `AddPermission` は、管理イベントとして分類され、 CloudTrail デフォルトでログインされます。などの Amazon SQS キューで実行される大量のオペレーションである Amazon SQS APIs `SendMessage` は、データイベントとして分類され、 でオプトインした後に ログに記録されます CloudTrail。

が CloudTrail 収集する情報を使用して、Amazon SQS API への特定のリクエスト、リクエスタの IP アドレスまたは ID、およびリクエストの日時を特定できます。 CloudTrail 証跡 を設定すると、Amazon S3 バケットに CloudTrail イベントを継続的に配信できます AWS EventBridge。 CloudWatch 証跡を設定しない場合、管理イベントのイベント履歴は CloudTrail コンソールのイベントでのみ表示できます。詳細については、AWS CloudTrail ユーザーガイドの「Overview for Creating a Trail」を参照してください。

の管理イベント CloudTrail

Amazon SQS は以下の API アクションを管理イベントとして記録します。

- `AddPermission`
- `CreateQueue`
- `CancelMessageMoveTask`
- `DeleteQueue`
- `ListMessageMoveTasks`
- `PurgeQueue`
- `RemovePermission`
- `SetQueueAttributes`
- `StartMessageMoveTask`
- `TagQueue`
- `UntagQueue`

ログ記録では CloudTrail、次の Amazon SQS APIs はサポートされていません。

- `GetQueueAttributes`
- `GetQueueUrl`

- ListDeadLetterSourceQueues
- ListQueueTags
- ListQueues

のデータイベント CloudTrail

データイベントは、Amazon SQS キューとの間で Amazon SQS メッセージを送受信するなど、リソース上で、またはリソース内で実行されるリソース操作に関する情報を提供します。データイベントは、デフォルトで CloudTrail はログに記録されない大量のアクティビティです。API を使用して、SQS キューのデータイベント CloudTrail APIs アクションのログ記録を有効にできます。詳細については、「AWS CloudTrail ユーザーガイド」の「データイベントを口グ記録する」を参照してください。

では CloudTrail、高度なイベントセレクタを使用して、どの Amazon SQS API アクティビティを口グに記録して記録するかを決定できます。Amazon SQS データイベントを口グに記録するには、リソースタイプ `AWS::SQS::Queue` を含める必要があります。これを設定したら、`eventName` フィルタを使用して `SendMessage` イベントを追跡するなど、記録する特定のデータイベントを選択して、口ギング設定をさらに絞り込むことができます。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。

Amazon SQS データイベント:

- SendMessage
- SendMessageBatch
- ReceiveMessage
- DeleteMessage
- DeleteMessageBatch
- ChangeMessageVisibility
- ChangeMessageVisibilityBatch

追加の変更がイベントデータに適用されます。詳細については、「AWS CloudTrail の料金」を参照してください。

例: Amazon SQS CloudTrail の管理イベント

次の例は、サポートされている APIsの CloudTrail 口グエントリを示しています。

AddPermission

次の例は、AddPermissionAPI コールの CloudTrail ログエントリを示しています。


```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
         "type": "IAMUser",
         "principalId": "AKIAI44QH8DHBEXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Alice",
         "accountId": "123456789012",
         "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
         "userName": "Alice"
       },
       "eventTime": "2018-06-28T22:23:46Z",
       "eventSource": "sqs.amazonaws.com",
       "eventName": "AddPermission",
       "awsRegion": "us-east-2",
       "sourceIPAddress": "203.0.113.0",
       "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
 Firefox/24.0",
       "requestParameters": {
         "actions": [
           "SendMessage"
         ],
         "AWSAccountIds": [
           "123456789012"
         ],
         "label": "MyLabel",
         "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
       },
       "responseElements": null,
       "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
       "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
     }
   ]
 }
```


CreateQueue

次の例は、CreateQueueAPI コールの CloudTrail ログエントリを示しています。

```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
         "type": "IAMUser",
         "principalId": "AKIAI44QH8DHBEXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Alejandro",
         "accountId": "123456789012",
         "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
         "userName": "Alejandro"
       },
       "eventTime": "2018-06-28T22:23:46Z",
       "eventSource": "sqs.amazonaws.com",
       "eventName": "CreateQueue",
       "awsRegion": "us-east-2",
       "sourceIPAddress": "203.0.113.1",
       "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
 Firefox/24.0",
       "requestParameters": {
         "queueName": "MyQueue"
       },
       "responseElements": {
         "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
       },
       "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
       "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
     }
   ]
 }
```


DeleteQueue

次の例は、DeleteQueueAPI コールの CloudTrail ログエントリを示しています。


```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
         "type": "IAMUser",
         "principalId": "AKIAI44QH8DHBEXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Carlos",
```

```
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Carlos"
      },
      "eventTime": "2018-06-28T22:23:46Z",
      "eventSource": "sqs.amazonaws.com",
      "eventName": "DeleteQueue",
      "awsRegion": "us-east-2",
      "sourceIPAddress": "203.0.113.2",
      "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
 Firefox/24.0",
      "requestParameters": {
        "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
      },
      "responseElements": null,
      "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
      "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
    }
  ]
}
```

RemovePermission

次の例は、RemovePermissionAPI コールの CloudTrail ログエントリを示しています。

```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
         "type": "IAMUser",
         "principalId": "AKIAI44QH8DHBEXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Jane",
         "accountId": "123456789012",
         "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
         "userName": "Jane"
       },
       "eventTime": "2018-06-28T22:23:46Z",
       "eventSource": "sqs.amazonaws.com",
       "eventName": "RemovePermission",
       "awsRegion": "us-east-2",
       "sourceIPAddress": "203.0.113.3",
       "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
 Firefox/24.0",
```

```
      "requestParameters": {
        "label": "label",
        "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
      },
      "responseElements": null,
      "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
      "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
    }
  ]
}
```

SetQueueAttributes

次の例は、 の CloudTrail ログエントリを示していますSetQueueAttributes。

```
{
   "Records": [
     {
       "eventVersion": "1.06",
       "userIdentity": {
         "type": "IAMUser",
         "principalId": "AKIAI44QH8DHBEXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Maria",
         "accountId": "123456789012",
         "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
         "userName": "Maria"
       },
       "eventTime": "2018-06-28T22:23:46Z",
       "eventSource": "sqs.amazonaws.com",
       "eventName": "SetQueueAttributes",
       "awsRegion": "us-east-2",
       "sourceIPAddress": "203.0.113.4",
       "userAgent": "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101
 Firefox/24.0",
       "requestParameters": {
         "attributes": {
           "VisibilityTimeout": "100"
         },
         "queueUrl": "https://sqs.us-east-2.amazon.com/123456789012/MyQueue"
       },
       "responseElements": null,
       "requestID": "123abcde-f4gh-50ij-klmn-60o789012p30",
       "eventID": "0987g654-32f1-09e8-d765-c4f3fb2109fa"
     }
```

```
    ]
  }
```

例:Amazon SQS CloudTrail のデータイベント

以下は、Amazon SQS データ CloudTrail イベント APIs。 Amazon SQS

SendMessage

次の例は、 CloudTrail のデータイベントを示していますSendMessage。

```
    {
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/SessionName",
    "accountId": "123456789012",
    "accessKeyId": "ACCESS_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed",
        "accountId": "123456789012",
        "userName": "RoleToBeAssumed"
      },
      "attributes": {
        "creationDate": "2023-11-07T22:13:06Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2023-11-07T23:59:11Z",
  "eventSource": "sqs.amazonaws.com",
  "eventName": "SendMessage",
  "awsRegion": "ap-southeast-4",
  "sourceIPAddress": "10.0.118.80",
  "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": {
```

```
    "queueUrl": "https://sqs.ap-southeast-4.amazonaws.com/123456789012/MyQueue",
    "messageBody": "HIDDEN_DUE_TO_SECURITY_REASONS",
    "messageDeduplicationId": "MsgDedupIdSdk1ae1958f2-bbe8-4442-83e7-4916e3b035aa",
    "messageGroupId": "MsgGroupIdSdk16"
  },
  "responseElements": {
    "mD5OfMessageBody": "9a4e3f7a614d9dd9f8722092dbda17a2",
    "mD5OfMessageSystemAttributes": "f88f0587f951b7f5551f18ae699c3a9d",
    "messageId": "93bb6e2d-1090-416c-81b0-31eb1faa8cd8",
    "sequenceNumber": "18881790870905840128"
  },
  "requestID": "c4584600-fe8a-5aa3-a5ba-1bc42f055fae",
  "eventID": "98c735d8-70e0-4644-9432-b6ced4d791b1",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::SQS::Queue",
      "ARN": "arn:aws:sqs:ap-southeast-4:123456789012:MyQueue"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "sqs.ap-southeast-4.amazonaws.com"
  }
```


ReceiveMessage

次の例は、 CloudTrail のデータイベントを示していますReceiveMessage。


```
    {
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/SessionName",
    "accountId": "123456789012",
```

```
    "accessKeyId": "ACCESS_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed",
        "accountId": "123456789012",
        "userName": "RoleToBeAssumed"
      },
      "attributes": {
        "creationDate": "2023-11-07T22:13:06Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2023-11-07T23:59:24Z",
  "eventSource": "sqs.amazonaws.com",
  "eventName": "ReceiveMessage",
  "awsRegion": "ap-southeast-4",
  "sourceIPAddress": "10.0.118.80",
  "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": {
    "queueUrl": "https://sqs.ap-southeast-4.amazonaws.com/123456789012/MyQueue",
    "maxNumberOfMessages": 10
  },
  "responseElements": null,
  "requestID": "8b4d4643-8f49-52cd-a6e8-1b875ed54b99",
  "eventID": "f3f23ab7-b0a4-4b71-afc0-141209c49206",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::SQS::Queue",
      "ARN": "arn:aws:sqs:ap-southeast-4:123456789012:MyQueue"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
```

```
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "sqs.ap-southeast-4.amazonaws.com"
  }
}
```

DeleteMessageBatch

次の例は、 CloudTrail のデータイベントを示していますDeleteMessageBatch。

```
    {
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/SessionName",
    "accountId": "123456789012",
    "accessKeyId": "ACCESS_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed",
        "accountId": "123456789012",
        "userName": "RoleToBeAssumed"
      },
      "attributes": {
        "creationDate": "2023-11-07T22:13:06Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2023-11-07T23:59:24Z",
  "eventSource": "sqs.amazonaws.com",
  "eventName": "DeleteMessageBatch",
  "awsRegion": "ap-southeast-4",
  "sourceIPAddress": "10.0.118.80",
  "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": {
    "queueUrl": "https://sqs.ap-southeast-4.amazonaws.com/123456789012/MyQueue",
    "entries": [
```

```
      {
        "id": "0",
        "receiptHandle": "AQEBefxM1O4zyZGF87DehbRbmri91w2W7mMdD0GrBjQa8e/
hpb4RbXHPZ9tLBVleECbChQIE5NtaDuoZhZPOkTy0eN46EyRR4jXDzE3AlkbPlX1mA9f2fUuTrXx8aeCoCA3I3woNg3f
hlLS94tjAZqV2krc4BaC2pYgjyHWcW019HwIV8T/bjNMIeZoQwOM5V
+o9vHPfewz5QGr5SKpDo7uE7Umyk5n5CJZvcn1efp/
mrwtaCIb9M7cCQUYcZm2ZmZDnIO9XpGTAi3m2dQ0M83pnNh0nvDfpkHpoa+hX1TrUmxCupCWHJwA8HFJ1O/
CCJsodMNFthLBA9S57dkBZCsw41G8jAmgQ0MkvZ0UL5mg0OFQQd1Yrw0zvthjCgiwdzn0yXoMzxIZMBxkY14E4nVVZ7N
h8oRk2C7gByzg2kYJ0LnUvLJFT8DQE28JZppEC9klvrdR/BWiPT7asc="
      }
    ]
  },
  "responseElements": {
    "successful": [
      {
        "id": "0"
      }
    ],
    "failed": []
  },
  "requestID": "fe423091-5642-5ba5-9256-6d5587de52f1",
  "eventID": "88c8020d-d769-4985-8ecb-ee0b59acc418",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::SQS::Queue",
      "ARN": "arn:aws:sqs:ap-southeast-4:123456789012:MyQueue"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "sqs.ap-southeast-4.amazonaws.com"
  }
}
```

ChangeMessageVisibilityBatch

次の例は、 CloudTrail のデータイベントを示していますChangeMessageVisibilityBatch。


```
    {
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/SessionName",
    "accountId": "123456789012",
    "accessKeyId": "ACCESS_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed",
        "accountId": "123456789012",
        "userName": "RoleToBeAssumed"
      },
      "attributes": {
        "creationDate": "2023-11-07T22:13:06Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2023-11-07T23:59:01Z",
  "eventSource": "sqs.amazonaws.com",
  "eventName": "ChangeMessageVisibilityBatch",
  "awsRegion": "ap-southeast-4",
  "sourceIPAddress": "10.0.118.80",
  "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": {
    "visibilityTimeout": 0,
    "entries": [
      {
        "id": "0",
        "receiptHandle":
 "AQEB2M5cVYg5gslhWME6537hdjcaPnOYPA5M0W460TTb0DzPle631yPWm8qxd4O1hDj/
B4ntTMnsgBTa95t14tNx7Vn96jKJ5rIoZ7iI8TRmkT1caKodKIPs8w9yndZq50c2FPQxtyH+2L3UHf/
abV3szqVWXOLZR4PwX8zZkWVQGNCNnY2q2lGCG586F8QwvrOFYoXNwB8ymd1t77e1PDPknq1Io3JFuzkEsndkkETy4fV
```

```
l5PHX17nXxaC+DURVlMPXOuSFACGmWqAoyk50HKwGOjLQgpySL/
TcnQXClvFq8kNXGwyVzJsbwHpOHxI7oce69vaD6DaWFP75d3hx+PJeG9pauQCKzVP3skt3Hw/
zDC7YfKcALD3aCwMmeNDwT3w0BUG6XZdG5lYhtFtTQYV7YuS3i/
Jh3HShGbtm07JKOEFiPkxv2+XNaAX3gFEpbng6zamTanfyMXCJIiglAEqiyWHQ=",
        "visibilityTimeout": 2271
      }
    ],
    "queueUrl": "https://sqs.ap-southeast-4.amazonaws.com/123456789012/MyQueue"
  },
  "responseElements": {
    "successful": [
      {
        "id": "0"
      }
    ]
  },
  "requestID": "d49ab65f-9dc7-54b8-875c-eb9b4c42988b",
  "eventID": "ca16c8c2-c4ba-4eb5-a54c-e650a10266d4",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::SQS::Queue",
      "ARN": "arn:aws:sqs:ap-southeast-4:123456789012:MyQueue"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "sqs.ap-southeast-4.amazonaws.com"
  }
}
```


を使用した Amazon SQS キューのモニタリング CloudWatch

Amazon SQS と Amazon CloudWatch は統合されているため、 CloudWatch を使用して Amazon SQS キューのメトリクスを表示および分析できます。キューのメトリクスは、Amazon SQS コン

ソール、コンソール、CloudWatch 、AWS CLIまたは CloudWatch API を使用して表示および分析できます。Amazon SQS メトリクスの CloudWatch アラームを設定することもできます。

CloudWatch Amazon SQS キューの メトリクスは自動的に収集され、 CloudWatch 1 分間隔で にプッシュされます。これらのメトリクスは、アクティブな になるための CloudWatch ガイドラインを満たすすべてのキューで収集されます。 CloudWatch は、メッセージが含まれている場合、またはアクションがキューにアクセスする場合、キューを最大 6 時間アクティブと見なします。

Amazon SQS キューが 6 時間以上非アクティブの場合、Amazon SQS サービスはスリープ状態と見なされ、 CloudWatch サービスへのメトリクスの配信を停止します。Amazon SQS キューが非アクティブであった期間中、Amazon SQS の CloudWatch メトリクスで欠落データまたはゼロを表すデータを視覚化することはできません。 Amazon SQS

> **Note**
>
> - Amazon SQS キューは、キューに対して API を呼び出すユーザーが承認されず、リクエストが失敗した場合にアクティブ化できます。
> - Amazon SQS コンソールは、キューのページが開かれると `GetQueueAttributes` API コールを実行します。`GetQueueAttributes` API リクエストはキューをアクティブ化します。
> - キューが非アクティブ状態からアクティブ化されると、 CloudWatch メトリクスで最大 15 分の遅延が発生します。
> - で報告された Amazon SQS メトリクスには料金はかかりません CloudWatch。これらは Amazon SQSサービスの一部として提供されます。
> - CloudWatch メトリクスは、標準キューと FIFO キューの両方でサポートされています。

トピック

- Amazon SQS の CloudWatch メトリクスへのアクセス
- Amazon SQS メトリクスの CloudWatch アラームの作成
- Amazon SQS で使用できる CloudWatch メトリクス

Amazon SQS の CloudWatch メトリクスへのアクセス

Amazon SQS と Amazon CloudWatch は統合されているため、 CloudWatch を使用して Amazon SQS キューのメトリクスを表示および分析できます。キューのメトリクスは、Amazon SQS コン

ソール、コンソール、CloudWatch 、AWS CLIまたは CloudWatch API を使用して表示および分析できます。Amazon SQS メトリクスの CloudWatch アラームを設定することもできます。

Amazon SQSコンソール

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、メトリクスにアクセスするキューのチェックボックスを選択 (オンに) します。最大10個のキューのメトリクスを表示できます。
3. モニタリングタブを選択します。

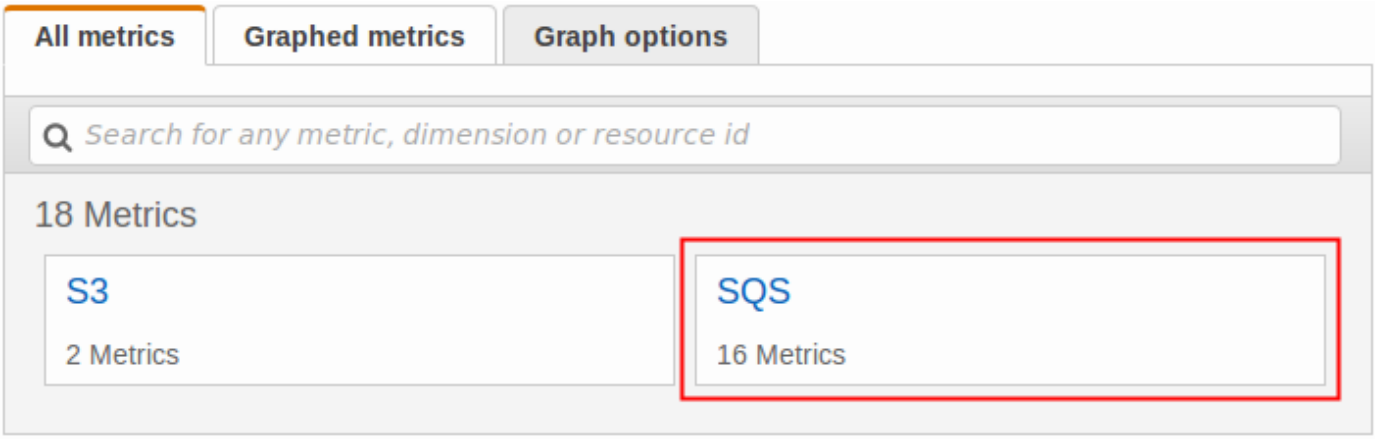
   さまざまなグラフが [SQS metrics] セクションに表示されます。
4. 特定のグラフが表す内容を理解するには、目的のグラフの横の

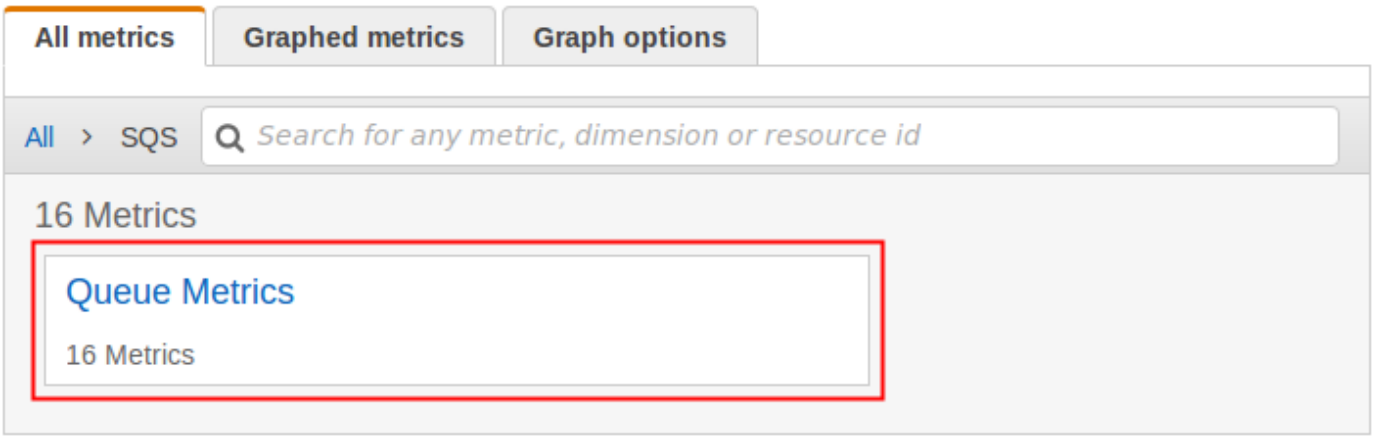
   にマウスを移動するか、「Amazon SQS で使用できる CloudWatch メトリクス」を参照してください。
5. すべてのグラフの時間範囲を同時に変更するには、[Time Range] で、目的の時間範囲 (たとえば、[Last Hour]) を選択します。
6. 個別のグラフの追加の統計を表示するには、そのグラフを選択します。
7. CloudWatch 詳細のモニタリングダイアログボックスで、統計 を選択します (例: Sum) 。サポートされている統計のリストについては、「Amazon SQS で使用できる CloudWatch メトリクス」を参照してください。
8. 個別のグラフで表示される時間の範囲と間隔を変更するには (たとえば、過去 5 分間ではなく過去 24 時間の範囲を表示したり、5 分ごとではなく 1 時間ごとの期間を表示するなど)、グラフのダイアログボックスが表示された状態で、[Time Range] の目的の時間範囲 (たとえば、[Last 24 Hours]) を選択します。[Period] で、指定された時間範囲 (たとえば、[1Hour]) 内で目的の期間を選択します。グラフの表示を終了するときは、[Close] を選択します。
9. (オプション) 追加 CloudWatch 機能を使用するには、モニタリングタブで、すべてのCloudWatchメトリクスを表示 を選択し、Amazon CloudWatch コンソール手順の指示に従います。

Amazon CloudWatch コンソール

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. [SQS] メトリクスの名前空間を選択します。

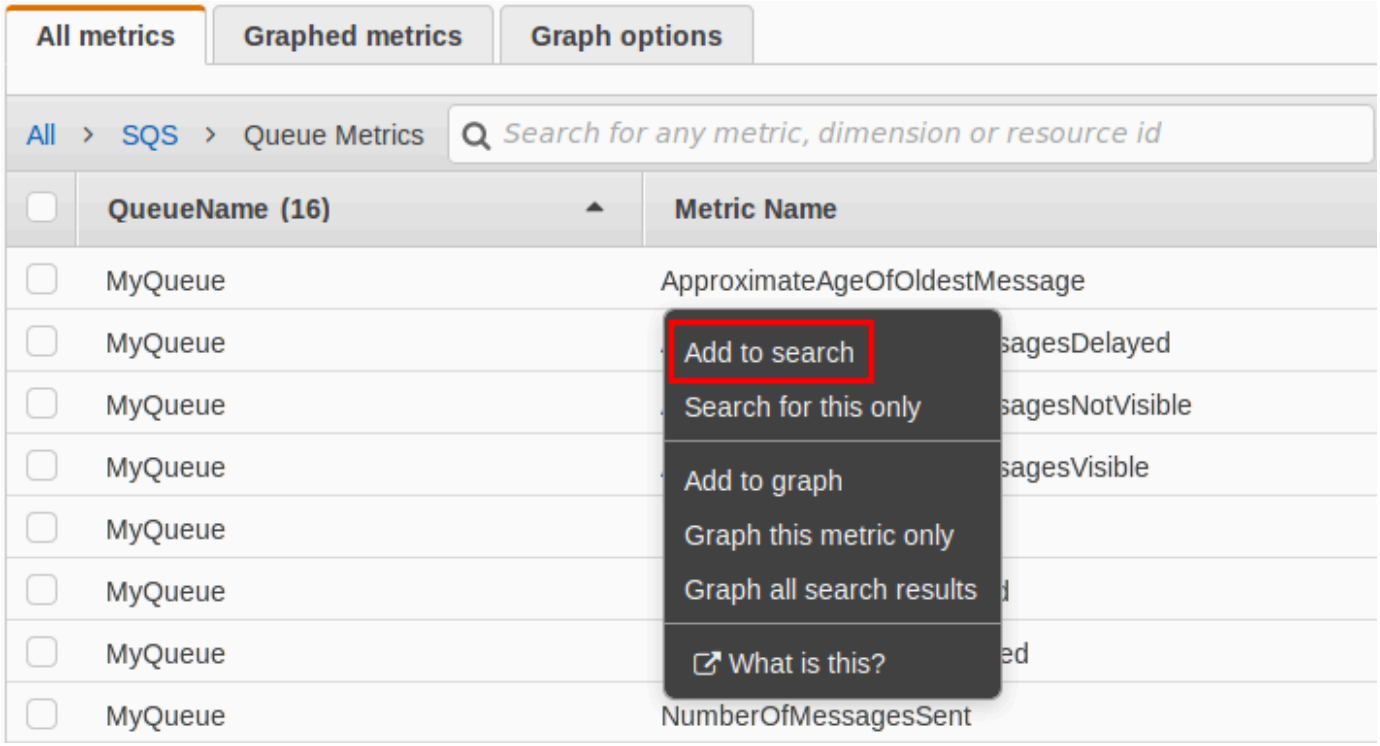


4. [Queue Metrics] メトリクスディメンションを選択します。



5. これで、Amazon SQSメトリクスを調べることができるようになりました。

   - メトリクスを並べ替えるには、列見出しを使用します。
   - メトリクスをグラフ表示するには、メトリクスの横にあるチェックボックスを選択します。
   - メトリクスでフィルタするには、メトリクスの名前を選択し、[Add to search] を選択します。



詳細および追加オプションについては、「Amazon ユーザーガイド」の「メトリクスのグラフ化」および「Amazon ダッシュボードの使用」を参照してください。 CloudWatch CloudWatch

AWS Command Line Interface

を使用して Amazon SQS メトリクスにアクセスするには AWS CLI、 `get-metric-statistics` コマンドを実行します。

詳細については、「Amazon ユーザーガイド」の「メトリクスの統計を取得する」を参照してください。 CloudWatch

CloudWatch API

CloudWatch API を使用して Amazon SQS メトリクスにアクセスするには、`GetMetricStatistics`アクションを使用します。

詳細については、「Amazon ユーザーガイド」の「メトリクスの統計を取得する」を参照してください。 CloudWatch

Amazon SQS メトリクスの CloudWatch アラームの作成

CloudWatch では、メトリクスのしきい値に基づいてアラームをトリガーできます。たとえば、NumberOfMessagesSent メトリクスのアラームを作成できます。たとえば、1時間以内に100件以上のメッセージが MyQueue キューに送信された場合、Eメール通知が送信されます。詳細については、「Amazon ユーザーガイド」の「Amazon CloudWatch アラームの作成」を参照してください。 CloudWatch

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. [Alarms]、[Create Alarm] の順に選択します。
3. [アラームの作成] ダイアログボックスの [メトリクスの選択] セクションで、[メトリクスの参照]、[SQS] を選択します。
4. SQS > キューメトリクス で、アラームを設定する QueueNameとメトリクス名を選択し、次へ を選択します。 使用可能なメトリクスのリストについては、Amazon SQS で使用できる CloudWatch メトリクス を参照してください。

   次の例では、NumberOfMessagesSentキューのMyQueueメトリクスのアラームが選択されています。送信されたメッセージの数が100を超えるとアラームがトリガーされます。
5. [アラームの作成] ダイアログボックスの、[アラームの定義] セクションで、次の操作を行います。

   a. [アラームのしきい値] で、アラームの [名前] と [説明] を入力します。
   b. [is] を [> 100] に設定します。
   c. [for] を [1 out of 1 datapoints (1つのデータポイントにつき 1 つ)] に設定します。
   d. [アラームのプレビュー] で、[間隔] を [1時間] に設定します。
   e. [統計] を [スタンダード]、[合計] に設定します。
   f. [アクション] の [アラームが次の時] に、[状態: 警告] を設定します。

      アラームがトリガーされたときに通知 CloudWatch を送信する場合は、既存の Amazon SNS トピックを選択するか、新しいリストを選択し、E メールアドレスをカンマで区切って入力します。

> **Note**
>
> 新しいAmazon SNSトピックを作成する場合、E 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メールアドレスが検証される前にアラーム状態が変更されると、通知は配信されません。

6. [アラームの作成] を選択します。

   アラームが作成されます。

Amazon SQS で使用できる CloudWatch メトリクス

Amazon SQS は、次のメトリクスを に送信します CloudWatch。

> **Note**
>
> スタンダードキューの場合、Amazon SQSの分散アーキテクチャのため、結果は概算になります。ほとんどの場合、カウントはキュー内の実際のメッセージ数に近い数値になります。FIFOキューの場合、結果は正確です。

Amazon SQSメトリクス

AWS/SQS 名前空間には、次のメトリクスが含まれます。

メトリクス	説明
ApproximateAgeOfOldestMessage	キューで最も古い削除されていないメッセージのおおよその経過期間。 **Note** • メッセージを3回 (またはそれ以上) 受信して処理しないと、メッセージはキューの後ろに移動され、Approxima

メトリクス	説明
	teAgeOfOldestMessage メトリクスは3回以上受信されていない2番目に古いメッセージを指します。このアクションは、キューに再処理ポリシーがある場合でも発生します。 • 1つのポイズンピルメッセージ (複数回受信されても削除されないメッセージ) がこのメトリクスを歪める可能性があるため、ポイズンピルメッセージが正常に消費されるまで、ポイズンピルメッセージの経過時間はメトリクスに含まれません。 • キューに再処理ポリシーがある場合、設定された最大受信数を超えると、メッセージはデッドレターキューに移動されます。メッセージがデッドレターキューに移動されると、デッドレターキューの ApproximateAgeOfOldestMessage メトリクスは、メッセージがデッドレターキューに移動された時間を表します (メッセージが送信された元の時間ではありません)。 • FIFO キューの場合、FIFO 順序の保証が破られるため、メッセージがキューの後ろに移動されることはありません。DLQ が設定されている場合、メッ

メトリクス	説明
	セージは代わりに DLQ に送られます。それ以外の場合、正常に削除される、または有効期限が切れるまで、メッセージグループがブロックされます。 レポート基準:キューがアクティブな場合、負以外の値が報告されます 単位: 秒 有効な統計:平均、最小、最大、合計、データサンプル (Amazon SQS コンソールにサンプル数として表示されます)
`ApproximateNumberOfMessagesDelayed`	遅延が発生したため、すぐに読み取ることのできない、キューのメッセージ数。これは、キューが遅延キューとして設定されている場合、またはメッセージが遅延パラメータとともに送信された場合に発生することがあります。 レポート基準:キューがアクティブな場合は負以外の値が報告されます。 単位:個 有効な統計:平均、最小、最大、合計、データサンプル (Amazon SQSコンソールにサンプル数として表示されます)

メトリクス	説明
ApproximateNumberOfMessagesNotVisible	処理中のメッセージの数。メッセージがクライアントに送信されたが、まだ削除されていない場合、または表示期限に達していない場合、メッセージは処理中とみなされます。 レポート基準:キューがアクティブな場合は負以外の値が報告されます。 単位:個 有効な統計:平均、最小、最大、合計、データサンプル (Amazon SQSコンソールにサンプル数として表示されます)
ApproximateNumberOfMessagesVisible	処理するメッセージの数。 レポート基準:キューがアクティブな場合は負以外の値が報告されます。 単位:個 有効な統計:平均、最小、最大、合計、データサンプル (Amazon SQSコンソールにサンプル数として表示されます) 処理するメッセージの数に制限はありませんが、このバックログには保存期間を設けることができます。

メトリクス	説明
NumberOfEmptyReceives 1	メッセージを返さなかった ReceiveMessage API 呼び出しの数。 レポート基準:キューがアクティブな場合は負以外の値が報告されます。 単位:個 有効な統計:平均、最小、最大、合計、データサンプル (Amazon SQSコンソールにサンプル数として表示されます)

メトリクス	説明
NumberOfMessagesDeleted 1	キューから削除されたメッセージの数。 レポート基準:キューがアクティブな場合は負以外の値が報告されます。 単位:個 有効な統計:平均、最小、最大、合計、データサンプル (Amazon SQSコンソールにサンプル数として表示されます) Amazon SQSは、NumberOfMessagesDeleted 有効なものを使用するすべての削除オペレーションのメトリック受信ハンドル、重複した削除を含む。を発行します。以下の場合は、NumberOfMessagesDeleted メトリクスの値が予想より高くなる場合があります。 • 同じメッセージの異なる受信ハンドルごとにDeleteMessage アクションを呼び出した場合: メッセージを処理する前に可視性タイムアウトが期限切れになると、メッセージは再び他のコンシューマーが処理して削除できるようになり、NumberOfMessagesDeleted メトリクスの値が増えます。 • 同じ受信ハンドルでDeleteMessage アクションを呼び出した場合: メッセージを処理して削除した後で同じ受信ハンドルを使用して再びDeleteMessage アクションを呼び出すと、成功ステータスが

メトリクス	説明
	返され、NumberOfMessagesDeleted メトリクスの値が増えます。
NumberOfMessagesReceived 1	ReceiveMessage アクションへの呼び出しで返されたメッセージの数。 レポート基準:負以外の値が報告されますキューがアクティブな場合。 単位:個 有効な統計:平均、最小、最大、合計、データサンプル (Amazon SQSコンソールにサンプル数として表示されます)
NumberOfMessagesSent 1	キューに追加されたメッセージの数。 手動でデッドレターキューに送信したメッセージは、NumberOfMessagesSent メトリクスによってキャプチャされます。ただし、処理が失敗したためにメッセージがデッドレターキューに送信された場合、このメトリクスではキャプチャされません。したがって、NumberOfMessagesSent と NumberOfMessagesReceived の値が異なることもあります。 レポート基準:キューがアクティブな場合は負以外の値が報告されます。 単位:個 有効な統計:平均、最小、最大、合計、データサンプル (Amazon SQSコンソールにサンプル数として表示されます)

メトリクス	説明
SentMessageSize [1]	キューに追加されたメッセージのサイズ。 レポート基準:負以外の値が報告されますキューがアクティブな場合。 単位:バイト 有効な統計: 平均、最小、最大、合計、データサンプル (Amazon SQSコンソールにサンプル数として表示されます) **Note** SentMessageSize は、少なくとも 1 つのメッセージが対応するキューに送信されるまで、CloudWatch コンソールに使用可能なメトリクスとして表示されません。

[1]これらのメトリックはサービスの観点から計算され、再試行を含めることができます。これらのメトリクスの絶対値に依存したり、現在のキューの状態を見積もったりしないでください。

Amazon SQS メトリクスのディメンション

Amazon SQS が送信する唯一のディメンションは CloudWatch ですQueueName。つまり、使用可能なすべての統計がQueueNameによってフィルタリングされます。

Amazon SQSのコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、コンプライアンスプログラムAWS のサービス による対象範囲内のコンプライアンスプログラムを参照し、関心のあるコンプライアンスプログラムを選択します。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。

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 AWS Artifact。詳細については、「 でのレポートのダウンロード AWS Artifact」の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、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。 は、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、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 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ — このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

> **Note**
>
> すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、「HIPAA 対応サービスのリファレンス」を参照してください。

- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド — コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス 、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「Security Hub のコントロールリファレンス」を参照してください。
- Amazon GuardDuty – これにより AWS アカウント、不審なアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワークで義

務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。

- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

Amazon SQSの耐障害性

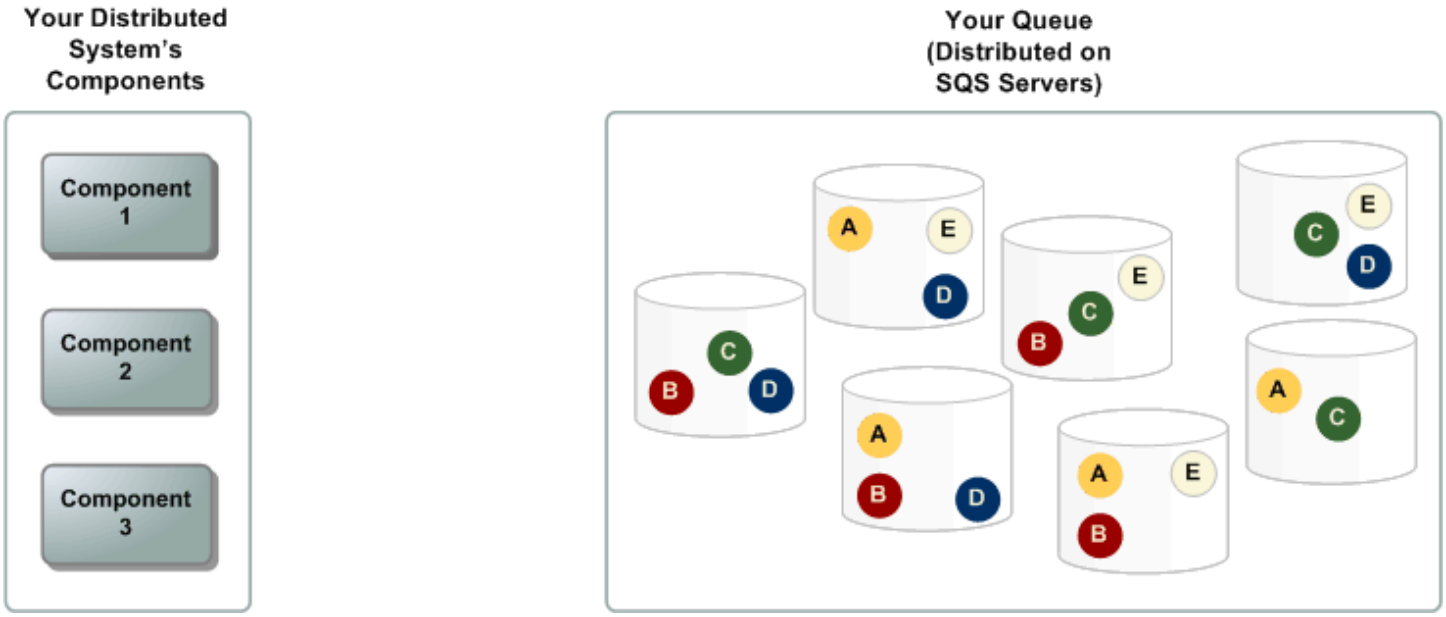
AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、高冗長ネットワークで接続された複数の物理的に分離されたアベイラビリティーゾーンと分離されたアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。 AWS リージョンと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。

Amazon SQS は、 AWS グローバルインフラストラクチャに加えて、分散キューを提供します。

分散キュー

分散メッセージングシステムには 3 つの主要部分 (分散システムのコンポーネント、キュー (Amazon SQS サーバーで分散)、およびキューのメッセージ) があります。

次のシナリオでは、システムには複数のプロデューサー (キューにメッセージを送信するコンポーネント) とコンシューマー (キューからメッセージを受信するコンポーネント) があります。キュー (A から E までのメッセージを保持) は、複数のAmazon SQS サーバー全体にまたがって冗長的にメッセージを保存します。



Amazon SQS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 SQS は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」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の手順で保護されています。

が AWS 公開した API アクションを使用して、ネットワーク経由で Amazon SQS 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 SQS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 SQSポリシーを使用して、特定の AmazonVPC 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 SQS キューへの 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、「例5:VPC エンドポイントからではない場合はアクセスを拒否する」を参照してください。

Amazon SQSのセキュリティベストプラクティス

AWS にはAmazon SQSの多くのセキュリティ機能が用意されています。これらは、独自のセキュリティポリシーのコンテキストで確認する必要があります。以下に、Amazon SQS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。

> **Note**
>
> 提供される特定の実装ガイダンスは、一般的なユースケースと実装用です。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。

トピック

キューがパブリックアクセスされていないことを確認する

インターネット上の誰かが Amazon SQS キューを読み書きできるように明示的に要求しない限り、キューがパブリックにアクセスできない (世界中の全員または認証された AWS ユーザーがアクセス可能) ことを確認する必要があります。

- `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

アクセス許可を付与する場合、アクセス許可を受け取るユーザー、アクセス許可の対象となるキュー、およびこれらのキューに対して許可する特定の APIアクションを決定します。最小権限を実

装することは、セキュリティリスクを軽減し、エラーや悪意のあるインテントの影響を軽減するために重要です。

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。これで、セキュリティポリシーの組み合わせを使用して実装できます。

Amazon SQSはプロデューサー-コンシューマーモデルを使用し、次の3種類のユーザーアカウントアクセスを使用します。

- 管理者-キューの作成、変更、削除にアクセスします。管理者は、キューポリシーも制御します。
- プロデューサー-キューへのメッセージの送信にアクセスします。
- コンシューマー-キューからのメッセージの受信および削除にアクセスします。

詳細については、次のセクションを参照してください。

- Amazon SQSでの Identity and Access Management
- Amazon SQS (Amazon SQS)、 API のアクセス権限: アクションとリソースのリファレンスについて
- Amazon SQSアクセスポリシー言語を使用したカスタムポリシーの使用

Amazon SQSアクセスを必要とするアプリケーションと AWS サービスに IAM ロールを使用する Amazon SQS

Amazon EC2 などのアプリケーションや AWS サービスが Amazon SQS キューにアクセスするには、 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ローテーションされないため、 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。

IAM ロールを使用して、Amazon SQS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。ロールを使用する場合、長期的な認証情報 (ユーザー名、パスワード、アクセスキーなど) を などの EC2 インスタンスまたは AWS サービスに配布する必要はありません AWS Lambda。代わりに、ロールは、アプリケーションが他の AWS リソースを呼び出すときに使用できる一時的なアクセス許可を提供します。

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 SQS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、アクセスする際にメッセージを復号します。SSE は で管理されるキーを使用します AWS Key Management Service。リクエストが認証され、お客様がアクセス許可を持っていれば、キュー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、「Amazon SQS での保管時の暗号化」および「Amazon SQS キー管理」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

HTTPS (TLS) を使用しない場合、ネットワークベースの攻撃者は、 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または操作できます man-in-the-middle。キューポリシーの `aws:SecureTransport` 条件を使用して、HTTPS(TLS) 経由の暗号化された接続のみを許可し、リクエストに SSL の使用を強制します。

VPC エンドポイントを使用して Amazon SQSにアクセスすることを検討する場合には

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キューがある場合は、VPC エンドポイントを使用して、特定の VPC 内のホストへのアクセスのみをキューに入れます。キュー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からのキューへのアクセスを制御できます。

Amazon SQS 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。
- キューポリシーを使用して、どのVPCまたは VPC エンドポイントがキューにアクセスできるかを制御できます。

詳細については、「Amazon SQSのAmazon Virtual Private Cloud エンドポイント」および「Amazon SQS用のAmazon 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。

Amazon SQSの関連リソース

次の表は、本サービスを利用する際に役立つと思われる関連リソースの一覧です。

リソース	説明
Amazon Simple Queue Service API リファレンス	アクション、パラメータ、データタイプに関する説明と、サービスから返されるエラーのリスト。
AWS CLI コマンドリファレンスで見る Amazon SQS	キューの操作に使用できる AWS CLI コマンドの説明。
のリージョンとエンドポイント	Amazon SQSのリージョンとエンドポイントに関する情報
製品ページ	Amazon SQS に関する情報の主要なウェブページ。
フォーラム	Amazon SQSに関連する技術的な質問について話し合うデベロッパーのためのコミュニティベースのフォーラムです。
AWS プレミアムサポート情報	AWS インフラストラクチャサービスでのアプリケーションの構築と実行に役立つ、1 対 1 の迅速な対応をサポートするサポートチャネルである AWS 、プレミアムサポートに関する情報の主要なウェブページ。

ドキュメント履歴

次の表は、 Amazon Simple Queue Service デベロッパーガイドの2019年1月以降の重要な変更点をまとめたものです。このドキュメントの更新に関する通知については、「RSS フィード」を参照してください。

サービス機能は、サービスが利用可能な AWS リージョンに段階的にロールアウト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の可用性、および更新に関する通知のサブスクライブについては、「 の最新情報 AWS」を参照してください。

変更	説明	日付
AWS JSON プロトコル	AWS JSON プロトコルを使用して API リクエストを行います。	2023 年 7 月 27 日
Amazon SQS の AWS マネージドポリシーとこれらのポリシーの更新について説明する新しいセクション	Amazon SQS には、特定のソースキューにある最新のメッセージ移動タスク (最大 10 個) を一覧表示できる新しいアクションが追加されました。このアクションは `ListMessageMoveTasks` API 演算に関連付けられています。	2023 年 6 月 7 日
API を使用したデッドレターキューのリドライブ	Amazon SQS API を使用してデッドレターキューのリドライブを設定します。	2023 年 6 月 7 日
Amazon SQS 用 ABAC	柔軟でスケーラブルなアクセス許可が行える、キューのタグを使用した属性ベースのアクセス制御 (ABAC)。	2022 年 11 月 10 日

FIFO の高スループット上限の引き上げ	商業リージョンの FIFO 高スループットモードのデフォルトクォータの引き上げと、FIFO 高スループットドキュメントの最適化を行いました。	2022 年 10 月 20 日
デフォルトのサーバー側の暗号化が利用可能	デフォルトで SQS 所有の暗号化 (SSE-SQS) を使用するサーバー側の暗号化。	2022 年 9 月 26 日
Amazon SQS の混乱した代理の保護サポートが利用可能	混乱した代理の保護を使用すると、リクエストに新しいヘッダーを指定できます。そのヘッダーは、Amazon SQS マネージド SSE を使用する際に KMS ポリシーの条件と照合されます。	2021 年 12 月 29 日
マネージド SSE が利用可能	Amazon SQS マネージド SSE (SSE-SQS)は、Amazon SQS 所有の暗号化キーを使用して、メッセージキューを介して送信される機密データを保護する、マネージドサーバー側の暗号化です。	2021 年 11 月 23 日
デッドレターキューのリドライブが利用可能	Amazon SQS では、スタンダードキューに対してデッドレターキューのリドライブをサポートします。	2021 年 11 月 10 日

FIFOキューのメッセージに対する高スループットが利用可能です。	Amazon SQS FIFOキューに対する高スループットは、FIFOキューの1 秒あたりのトランザクション(TPS)数を増加することができます。スループットクォータの詳細については、「メッセージに関連するクォータ」を参照してください。	2021年5月27日
FIFOキューのメッセージに対する高スループットはプレビューリリースが利用可能です	Amazon SQS FIFOキューに対する高スループットはプレビューリリースであり、変更される可能性があります。この機能は、FIFO キューのメッセージに対して、1 秒あたりのトランザクション (TPS) の数を増加することができます。スループットクォータの詳細については、「メッセージに関連するクォータ」を参照してください。	2020年12月17日
新Amazon SQSコンソールのデザイン	開発および本番ワークフローを簡素化するために、Amazon SQS コンソールには新しいユーザーエクスペリエンスがあります。	2020年7月8日
Amazon SQS は listQueues と のページ分割をサポートしています listDeadLetterSourceQueues	listQueues または listDeadLetterSourceQueues リクエストから返す結果の最大数を指定できます。	2020年6月22日

Amazon SQS は、 GovCloud (米国) AWS リージョンを除くすべてのリージョンで 1 分間の AWS Amazon CloudWatch メトリクスをサポートします。	Amazon SQS の 1 分 CloudWatch メトリクスは、リージョンを除くすべての AWS GovCloud (US) リージョンで使用できます。	2020年1月9日
Amazon SQS が 1 分間の CloudWatch メトリクスをサポート	Amazon SQS の 1 分 CloudWatch メトリクスは、現在、米国東部 (オハイオ) 、欧州 (アイルランド) 、欧州 (ストックホルム) 、アジアパシフィック (東京) の各リージョンでのみ利用できます。	2019 年 11 月 25 日
AWS Lambda Amazon SQS FIFO キューの トリガーが利用可能に	Lambda関数をトリガーとしてFIFOキューへの着信メッセージを設定できます。	2019年11月25日
Amazon SQS に対する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。	Amazon SQSに対するSSEは、中国リージョンで利用可能です。	2019年11月13日
FIFOキューは、中東 (バーレーン)リージョンで利用可能です。	FIFOキューは、中東 (バーレーン)リージョンで利用可能です。	2019年10月10日
Amazon SQS 用の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントは、 AWS GovCloud (米国東部) および AWS GovCloud (米国西部) リージョンで利用できます。 Amazon SQS	(米国東部) および AWS GovCloud (AWS GovCloud 米国西部) リージョンの Amazon VPC から Amazon SQS キューにメッセージを送信できます。	2019年9月5日

Amazon SQS では、メッセージシステム属性 AWS X-Ray を使用してキューのトラブルシューティングを行うことができます。	X-Rayを使用して、Amazon SQS キュー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。このリリースでは、MessageSystemAttribute リクエストパラメータにSendMessage とSendMessageBatch APIオペレーション、ReceiveMessage API オペレーションのAWSTraceHeader 属性とMessageSystemAttributeValue データタイプが（ Amazon SQS)を通してX-Rayトレースヘッダーへ送信できる）が追加されます。	2019年8月28日
Amazon SQSキュー作成時にタグ付けが可能になります。	単一の Amazon SQS API コール、 AWS SDK 関数、または AWS Command Line Interface （ AWS CLI) コマンドを使用して、キューを同時に作成し、タグを指定できます。さらに、Amazon SQS は aws:TagKeys および aws:RequestTag AWS Identity and Access Management (IAM) キーをサポートしています。	2019年8月22日

Amazon SQS に対する一時キュークライアントは、現在利用可能です。	一時キューは、リクエストとレスポンスのような一般的なメッセージパターンを使用する場合に、開発時間と開発コストを削減するのに役立ちます。一時キュークライアントを使用すると、高スループットで費用対効果の高いアプリケーションマネージドの一時キューを作成ができます。	2019年7月25日
SSE for Amazon SQS が AWS GovCloud（米国東部）リージョンで利用可能に	Amazon SQS のサーバー側の暗号化 (SSE) は、 AWS GovCloud（米国東部）リージョンで利用できます。	2019年6月20日
FIFO キューは、アジアパシフィック (香港）、中国 (北京） AWS GovCloud 、（米国東部）、および AWS GovCloud（米国西部) の各リージョンで利用できます。	FIFO キューは、アジアパシフィック (香港）、中国 (北京） AWS GovCloud 、（米国東部）、および AWS GovCloud（米国西部) の各リージョンで利用できます。	2019年5月15日
Amazon SQSに対するAmazon VPC エンドポイントポリシーは利用可能です。	Amazon SQSに対するAmazon VPC エンドポイントポリシーを作成できます。	2019年4月4日
FIFOキューは、欧州(ストックホルム) および中国 (寧夏) リージョンで利用可能です。	FIFOキューは、欧州(ストックホルム) および中国 (寧夏) リージョンで利用可能です。	2019年3月14日

FIFO キューは、Amazon SQS が利用可能なすべてのリージョンで利用可能です。	FIFOキューは、米国東部（オハイオ）、米国東部（北バージニア）、米国西部（北カリフォルニア）、米国西部（オレゴン）、アジアパシフィック（ムンバイ）、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）、欧州（アイルランド）、欧州（ロンドン）、欧州（パリ）、および南米国（サンパウロ）の各リージョンで利用可能です。	2019年2月7日

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。